BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

METODOLOGÍA DE APRENDIZAJE SEMIAUTOMÁTICO PARA EL TRATAMIENTO DE DATOS BIOMÉDICOS: CASO DE ESTUDIO COVID-19

TESIS

PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
ICC. LUIS YAEL MÉNDEZ SÁNCHEZ

DIRECTOR DE TESIS:
DRA. MARÍA JOSEFA SOMODEVILLA GARCÍA

CO-DIRECTOR:
DRA. MARÍA DE LA CONCEPCIÓN PÉREZ DE CELIS HERRERO

PUEBLA, PUE. AGOSTO 2021

Dedicatoria

Dedico este trabajo primeramente a Dios, por permitirme el haber llegado a este momento tan importante de mi formación profesional, por darme la fortaleza para continuar cuando he estado a punto de caer, por iluminarme y darme la inteligencia necesaria para poder llevar a cabo el desarrollo de esta tesis.

A mis padres, Flora Méndez Sánchez y Juan De Jesús Merales, que con su esfuerzo y dedicación me ayudaron en todo mi proceso de formación profesional, tanto de licenciatura como ahora de maestría, gracias infinitas por siempre estar para apoyarme, por todos los regaños, consejos y palabras de aliento cuando sentía que no podía más y quería darme por vencido, gracias a ustedes siempre saqué fuerzas y convicción para seguir en el camino y así poder llegar a la meta. Gracias por esforzarse día con día en darme lo mejor, por creer siempre en mí y sobre todo por darme siempre su amor de una manera incondicional. Soy muy afortunado de tenerlos como padres, definitivamente Dios me ha dado la mayor bendición al tenerlos a ustedes como los pilares de mi vida.

A mis hermanas Joana y Sarahí, por estar conmigo en este proceso, por todas las palabras de aliento que me dedicaron para seguir adelante, por apoyarme y tolerarme cuando sentía que no podía más y estaba con mi mal humor, gracias por confiar y por creer en mí en todo momento, porque aunque cada una tiene su manera de demostrar su cariño, sé que me quieren como yo a ellas; siempre serán parte fundamental en mi vida, porque el amor de hermanos puede expresarse de diferentes maneras, pero de lo que sí estoy seguro es que dura para siempre.

Y finalmente a mis abuelos Romualdo y Luisa, porque en vida siempre me brindaron su amor y apoyo incondicional, estoy seguro que desde el cielo están muy orgullosos de mí por todo lo que estoy logrando, por ver el profesionista en el que me he convertido y sé que en todo momento a pesar de no tenerlos físicamente están al pendiente de mí, siempre los llevaré presentes en mi corazón.

Luis Yael Méndez Sánchez

Agradecimientos

Mi más profundo agradecimiento al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo otorgado a través de la beca para estudios de Maestría durante todo este proceso de formación.

A la Benemérita Universidad Autónoma de Puebla y al posgrado de la Facultad de Ciencias de la Computación, por permitirme ser parte de su comunidad universitaria desde la licenciatura y llevar a cabo mis estudios de posgrado en mi etapa de formación profesional.

Agradezco a mi director de tesis, la Dra. María Josefa Somodevilla García, por todos los conocimientos compartidos, su valiosa guía, asesoramiento y paciencia para conmigo. Gracias por todas esas clases impartidas desde la licenciatura, gracias por su experiencia y profesionalismo, además por el aprendizaje en cuanto al ámbito profesional.

A mi novia Suri, quien es la persona que quiero en estos momentos, le agradezco por compartir conmigo la mayoría del tiempo durante esta etapa de mis estudios de Maestría, donde nunca faltó su amor y apoyo incondicial, dedicándome palabras de motivación para seguir adelante y no desistir cuando quería darme por vencido a causa del estrés, gracias por estar conmigo hasta en mis momentos difíciles tolerándome y siendo esa persona especial que me da felicidad.

A mis amigos de la maestría Paco y Juanito, por todo el apoyo que me brindaron para sacar adelante muchos trabajos, por todos sus conocimientos y siempre considerarme parte de este gran equipo de trabajo donde compartimos tanto aspectos escolares como de nuestra vida personal. Quiero agradecer también a mis demás amigos de la generación, Nalle, Beto, Daniel, Irving, Anel y Jairo, porque desde que nos conocimos siempre me brindaron su apoyo para trabajar en conjunto, por todas esas tardes de estudio para los exámenes y sobre todo gracias por tan bonita amistad.

Agradezco también a mis demás profesores de la maestría, ya que gracias a ellos he aprendido sobre las bases de datos, procesamiento de lenguaje natural, recuperación de información y web semántica, gracias a todos los conocimientos impartidos en sus clases, he logrado formarme como un mejor profesionista.

A mis amigos de Puebla (Beto, Toño, Lalo, Erick y Leonardo), con los cuales pude compartir grandes momentos desde mi etapa de licenciatura y aunque en esta ocasión no estuvimos juntos en la maestría, siempre han estado presentes en todo momento, brindándome su apoyo incondicional tanto moral como profesionalmente, por todas esas pláticas de motivación, porque desde que llegué a Puebla ustedes se convirtieron en mi familia.

A la Sra. Betty y a su familia, por acogerme desde que llegué a estudiar a Puebla, por ser como una madre para mí, ya que al estar lejos de casa con ella nunca me ha faltado cariño ni amor, siempre bríndandome su apoyo incondicional.

Gracias a todos por seguir ayudándome a formarme como persona y como profesionista, realmente este es un logro en conjunto del cual siempre les estaré eternamente agradecido.

Resumen

En este proyecto de tesis se propone desarrollar una metodología de aprendizaje semiautomático para generar conocimiento acerca del COVID-19 y poder contribuir con herramientas de software útiles, que permitan el manejo de la información referente a esta pandemia.

El trabajo parte de la recopilación de datos de pacientes que han dado positivo a COVID-19, que es la enfermedad que ha originado esta pandemia a nivel mundial, para que con ello, se lleve a cabo el pre-procesamiento de la información y se obtenga una representación estructurada y homogénea de los datos, los cuales son tratados con tecnologías de Recuperación de Información, Big Data, Web Semántica, Procesamiento de Lenguaje Natural y Aprendizaje Automático. Junto con ello, en esta misma etapa se llevará a cabo el diseño e implementación de una base de datos en el DBMS conocido como MariaDB, el cual permite el uso de herramientas de su hómologo MySQL. Después, en la parte del procesamiento de la información se realizaron actualizaciones en los datos en la BD implementada en MariaDB para poder llevarla a una segunda implementación que será utilizando un motor de búsqueda basado en la nube llamado ElasticSearch. Este motor de búsqueda tiene la ventaja de permitir el almacenamiento y manejo de grandes volúmenes de información, para tener un conjunto de datos bastante completo y finalmente llevar a cabo el análisis de patrones en los datos mediante técnicas de aprendizaje automático y proporcionar información adicional y relevante en relación a la pandemia de COVID-19 que se vive en México.

Abstract

In this thesis project, it is proposed to develop a semi-automatic learning methodology to generate knowledge about COVID-19 and to be able to contribute with useful software tools that allow the management of information regarding this pandemic.

The work starts from the collection of data from patients who have tested positive for COVID-19, which is the disease that has caused this pandemic worldwide, so that with this, the pre-processing of the information is carried out and obtain a structured and homogeneous representation of the data, which is treated with Information Retrieval technologies, Big Data, Semantic Web, Natural Language Processing and Machine Learning. Along with this, in this same stage the design and implementation of a database in the DBMS known as MariaDB will be carried out, which allows the use of tools from its MySQL homolog. Later, in the information processing part, updates were made to the data in the DB implemented in MariaDB to be able to take it to a second implementation that will be using a cloud-based search engine called ElasticSearch. This search engine has the advantage of allowing the storage and handling of large volumes of information, to have a fairly complete data set and finally carry out the analysis of patterns in the data using machine learning techniques and provide additional and relevant information. in relation to the COVID-19 pandemic in Mexico.

ÍNDICE GENERAL

DEDICATORIA ..... **I**

AGRADECIMIENTOS ..... **II**

RESUMEN ..... **III**

ABSTRACT ..... **IV**

CAPÍTULO 1. INTRODUCCIÓN **1**

1.1 La epidemia de COVID-19 1
1.2 Problema de salud en México debido al COVID-19 2
1.2.1 Semáforo COVID-19 ..... *5*
1.3 Definición del problema 6
1.4 Justificación 7
1.5 Objetivos y preguntas de investigación 8
1.4 Alcances y limitaciones 9

CAPÍTULO 2. MARCO TEÓRICO **10**

2.1 Conceptos teóricos 10
2.1.1 Recuperación de información *10*
2.1.2 Ontologías *12*
2.1.3 Big Data *13*
2.1.4 Aprendizaje Automático *14*
2.2 Web Scraping 15
2.3 Python 15
2.4 BeautifulSoup 17
2.5 Gestores de Datos 17
2.5.1 ElasticSearch *19*
2.5.2 MariaDB *19*

CAPÍTULO 3. ESTADO DEL ARTE **21**

3.1 COVID-19 21
3.2 Aplicaciones de la ontología SNOMED-CT y CIDO Ontology 24
3.3 Aplicaciones en Procesamiento de Lenguaje Natural 24
3.4 Aplicaciones en Machine Learning 25
3.5 Análisis y limitantes 26

CAPÍTULO 4. METODOLOGÍA **27**

4.1 Fase 1: Búsqueda de datos COVID-19 28
4.1.1 Web Scraping para búsqueda de documentos médicos 28
4.1.2 Principal fuente de obtención de datos COVID-19 29
4.2 Fase 2: Pre-procesamiento de la información 30
4.2.1 Implementación en MariaDB *30*
4.3 Fase 3: Procesamiento de datos 35
4.3.1 Implementación en ElasticSearch *36*
4.3.2 Modificaciones en MariaDB *39*

4.3.3 Actualización de la base de datos en MariaDB *42*
4.3.4 Actualización de la base de datos en ElasticSearch *44*
4.4 Fase 4: Descubrimiento de patrones en los datos 46
4.4.1 Preparación del conjunto de datos *47*
4.4.2 Aplicación de técnicas de minería de datos *51*
4.4.3 Evaluación de la información *57*
CAPÍTULO 5. RESULTADOS 60
5.1 Base de datos en MariaDB 60
5.2 Base de datos en ElasticSearch 66
5.3 Resultados de la aplicación de técnicas de minería de datos 71
5.3.1 Análisis de parámetros *75*
5.3.2 Detalle de precisión por clase *75*
5.3.3 Matriz de confusión *76*
5.4 Obtención de la tasa de mortalidad y letalidad por entidad 77
CAPÍTULO 6. CONCLUSIONES 83
REFERENCIAS 85

ÍNDICE DE FIGURAS

Figura 1. Datos de COVID-19 en México 3
Figura 2. Porcentajes de casos confirmados 3
Figura 3. Porcentajes de acuerdo a comorbilidades principales 3
Figura 4. Gráfica de casos confirmados 4
Figura 5. Histograma de casos confirmados a nivel nacional 4
Figura 6. Histograma con rangos de edad y tipo paciente 5
Figura 7. Semáforo epidemiológico por estado en México 6
Figura 8. Medidas básicas de prevención 6
Figura 9. Metología propuesta para el desarrollo de este proyecto. 27
Figura 10. Fuentes de libre acceso para búsqueda de información COVID-19. 28
Figura 11. Web scrapper programado en Python para obtener artículos científicos. 28
Figura 12. Tablero de información COVID-19 en la página del Gobierno de México. 29
Figura 13. Conjunto de datos en formato CSV, obtenidos de la página del Gobierno de México. 30
Figura 14. Diagrama entidad - relación para la base de datos COVID-19. 30
Figura 15. Base de datos "covid19_pruebas" implementada en MariaDB. 31
Figura 16. Script realizado en Python para cargar los registros de pacientes con COVID-19 a la BD. 32
Figura 17. Archivo en formato CSV con la información que será procesada para insertar en la BD. 32
Figura 18. Consultas en SQL para la clasificación de acuerdo al tipo de paciente. 33
Figura 19. Script para la inserción de las enfermedades crónicas y sintomatologías que presenta cada paciente además del COVID-19. 33
Figura 20. Programa en Python que se encarga de realizar la obtención de cada uno de los pacientes registrados. 34
Figura 21. Código en Python que se encarga de realizar el update a la BD para insertar la ubicación de cada registro. 35
Figura 22. Herramientas de software con las que trabaja en conjunto ElasticSearch. 35
Figura 23. Script en lenguaje Logstash para insertar datos en el servidor de ElasticSearch. 36
Figura 24. Definición de tipos de dato por cada atributo y salida al servidor de ElasticSearch en lenguaje Logstash. 37
Figura 25. JSON creado por ElasticSearch para hacer la inserción de los datos dentro de su servidor. 37
Figura 26. Gestión de índices en la interfaz de Kibana. 38
Figura 27. Patrón de índice "datos-covid-pb" creado para trabajar con todos los datos insertados en ElasticSearch. 38
Figura 28. Base de datos de COVID-19 creada en ElasticSearch. 39
Figura 29. Consulta en SQL para clasificar a los pacientes por defunción. 39
Figura 30. Consulta en SQL para clasificar los pacientes en remisión. 39
Figura 31. Consulta en SQL para crear la tabla auxiliar "paciente_sin_ubicacion" en la BD. 40
Figura 32. Base de datos "covid19_pruebas" con nuevas tablas creadas. 40
Figura 33. Script en Python para calcular el número de días que tardaron en atenderse los pacientes en remisión. 40
Figura 34. Script que calcula el número de días de hospitalización que tardaron los pacientes antes de su defunción. 41

FIGURA 35. MUESTRA DE LA TABLA PACIENTES EN REMISIÓN PARA DISTINGUIR LA COLUMNA "DIAS_ATENDERSE". 41
FIGURA 36. MUESTRA DE LA TABLA DE PACIENTES DIFUNTOS, PARA DISTINGUIR LA COLUMNA "DIAS_HOSPITALIZADOS". 42
FIGURA 37. SCRIPT EN PYTHON ACTUALIZADO PARA CARGAR LOS REGISTROS DE PACIENTES CON COMORBILIDADES. 43
FIGURA 38. CONSULTAS EN SQL PARA INSERTAR LOS DATOS DE ACUERDO AL CRITERIO DE CLASIFICACIÓN. 43
FIGURA 39. BD "COVID19_ACT_COMORBILIDADES" CON LAS TABLAS E INFORMACIÓN ACTUALIZADA AL 31 DE MAYO DE 2021. 44
FIGURA 40. SE INDICA LA RUTA DEL PATH DONDE SE ENCUENTRA EL ARCHIVO CON LOS REGISTROS DE PACIENTES. 44
FIGURA 41. DEFINICIÓN DE CADA UNO DE LOS ATRIBUTOS EN LOGSTASH. 45
FIGURA 42. DEFINICIÓN DEL FORMATO ESPECÍFICO PARA LAS FECHAS QUE SE TIENEN COMO ATRIBUTOS. 45
FIGURA 43. DEFINICIÓN DEL TIPO DE DATO POR CADA UNO DE LOS ATRIBUTOS. 46
FIGURA 44. SALIDA AL SERVIDOR DE ELASTICSEARCH CON EL NUEVO NOMBRE DE LA BASE DE DATOS ACTUALIZADA. 46
FIGURA 45. TABLA GRUPOS_SECTOR EN LA BD DE COVID 19. 48
FIGURA 46. TABLA ENTIDADES_POR_REGION EN LA BD DE COVID 19 CON DATOS ACTUALIZADOS. 48
FIGURA 47. SCRIPT EN PYTHON PARA ASIGNAR A LOS PACIENTES EL NUEVO GRUPO DEL SECTOR AL QUE PERTENECEN. 48
FIGURA 48. FUNCIÓN EN PYTHON QUE SE ENCARGA DE ASIGNAR EL NUEVO ID DE LA ENTIDAD DE RESIDENCIA POR CADA PACIENTE REGISTRADO. 49
FIGURA 49. FUNCIÓN QUE CALCULA LOS DÍAS QUE TARDÓ EL PACIENTE EN ATENDERSE. 50
FIGURA 50. FUNCIÓN QUE CALCULA LOS DÍAS DE HOSPITALIZACIÓN DE LOS PACIENTES DIFUNTOS. 50
FIGURA 51. FUNCIÓN QUE ASIGNA EL ESTADO DEL PACIENTE A LOS DATOS. 51
FIGURA 52. CONSULTAS EN SQL PARA OBTENER LA MUESTRA DE PACIENTES PARA WEKA. 51
FIGURA 53. CONEXIÓN DE LA BASE DE DATOS CON WEKA. 52
FIGURA 54. DATOS PRESENTADOS EN WEKA PARA SU ANÁLISIS. 52
FIGURA 55. CONJUNTO DE DATOS DE ENTRENAMIENTO SIN ATRIBUTOS IRRELEVANTES. 53
FIGURA 56. OPCIONES DEL FILTRO "NUMERICTONOMINAL" EN WEKA. 54
FIGURA 57. OPCIONES DEL FILTRO "DISCRETIZE" PARA CONVERTIR LA EDAD A TIPO NOMINAL. 54
FIGURA 58. GRUPOS DE EDAD FORMADOS AL APLICAR EL FILTRO "DISCRETIZE" EN WEKA. 55
FIGURA 59. ATRIBUTOS QUE MEJOR PREDICE LA VARIABLE DE CLASE ESTADO_PACIENTE. 55
FIGURA 60. PRIMERA PARTE DEL ÁRBOL, CON EL TIPO DE PACIENTE IGUAL A AMBULATORIOS. 56
FIGURA 61. SEGUNDA PARTE DEL ÁRBOL, CON EL TIPO DE PACIENTE IGUAL A HOSPITALIZADOS. 56
FIGURA 62. OPCIONES DEL ALGORITMO J48 EN WEKA. 57
FIGURA 63. FUNCIÓN PARA OBTENER EL NÚMERO DE CASOS POSITIVOS POR ENTIDAD. 58
FIGURA 64. FUNCIÓN PARA OBTENER EL NÚMERO DE DEFUNCIONES POR ENTIDAD. 58
FIGURA 65. FUNCIÓN QUE CALCULA LA TASA DE MORTALIDAD DE LOS PACIENTES POR ENTIDAD. 58
FIGURA 66. FUNCIÓN QUE CALCULA LA TASA DE LETALIDAD POR CADA 100 PACIENTES POSITIVOS POR ENTIDADES. 59
FIGURA 67. TABLAS DE LA BASE DE DATOS DE COVID-19 ACTUALIZADA. 60
FIGURA 68. EJEMPLO DE LA TABLA "PACIENTE_BASE" CON ALGUNOS DE LOS ATRIBUTOS QUE CORRESPONDEN A LOS DATOS DE CADA PACIENTE. 61
FIGURA 69. REPRESENTACIÓN DE LA TABLA "GRUPOS_SECTOR". 62
FIGURA 70. REPRESENTACIÓN DE LA TABLA "ENTIDADES_POR_REGION". 62
FIGURA 71. TABLA PRESENTADA EN LA BD DE LOS SECTORES REGISTRADOS. 62
FIGURA 72. ENTIDADES DE LA REPÚBLICA MEXICANA REGISTRADAS EN LA BD. 63
FIGURA 73. REPRESENTACIÓN DE LA TABLA ENFERMEDADES EN LA BD. 63

FIGURA 74. REPRESENTACIÓN DE LA TABLA "PACIENTE_TIENE_ENFERMEDAD" EN LA BD. 63
FIGURA 75. INFORMACIÓN REGISTRADA EN LA TABLA CLASIFICACIÓN FINAL PARA ASIGNAR A LOS PACIENTES. 64
FIGURA 76. REPRESENTACIÓN DE LOS DATOS EN LA TABLA "PACIENTES_CLASIFICADOS" CON LOS NUEVOS ATRIBUTOS AGREGADOS. 64
FIGURA 77. ACTUALIZACIÓN DEL ID DE SECTOR PARA EL CONJUNTO DE DATOS. 65
FIGURA 78. ACTUALIZACIÓN DEL ID DE LA ENTIDAD DE RESIDENCIA POR LA REGIÓN A LA QUE PERTENECEN. 65
FIGURA 79. ALGUNOS DATOS DE LA MUESTRA DE PACIENTES POR REGIÓN PARA APLICAR MINERÍA DE DATOS. 66
FIGURA 80. GESTIÓN DE ÍNDICES CREADOS PARA LA BD DE COVID-19 EN ELASTICSEARCH. 66
FIGURA 81. PATRÓN DE ÍNDICE CREADO QUE SERÁ EL QUE CONTENGA TODO EL CONJUNTO DE DATOS. 67
FIGURA 82. GRÁFICA POR SEMANA DE LOS PACIENTES REGISTRADOS CON COVID-19 EN ELASTICSEARCH. 67
FIGURA 83. GRÁFICA DE PACIENTES REGISTRADOS POR MES. 68
FIGURA 84. FORMA EN QUE SE PRESENTAN LOS DATOS REGISTRADOS EN LA BD DE ELASTICSEARCH. 68
FIGURA 85. ESTRUCTURA DE UNO DE LOS REGISTROS QUE SE INSERTAN EN EL SERVIDOR DE ELASTICSEARCH. 69
FIGURA 86. FORMATO JSON PARA UNO DE LOS REGISTROS QUE SERÁN INSERTADOS EN ELASTICSEARCH. 70
FIGURA 87. ANÁLISIS DE LOS DATOS DESDE LA INTERFAZ DE KIBANA. 70
FIGURA 88. PRIMERA PARTE DEL ÁRBOL OBTENIDO CON EL FACTOR DE PODA EN 0.025. 71
FIGURA 89. SEGUNDA PARTE DEL ÁRBOL CON UN FACTOR DE PODA EN 0.025 Y EL TIPO DE PACIENTES ES HOSPITALIZADOS. 71
FIGURA 90. RESULTADOS DEL MODELO CON LA APLICACIÓN DEL ALGORITMO J48. 72
FIGURA 91. ÁRBOL BINARIO COMO RESULTADO DEL ALGORITMO J48 AL EVALUAR EL CONJUNTO DE DATOS DE ENTRENAMIENTO. 73
FIGURA 92. GRÁFICA POR RANGOS DE EDAD QUE MUESTRA LOS PACIENTES RECUPERADOS Y DIFUNTOS. 74
FIGURA 93. GRÁFICA QUE MUESTRA EL TOTAL DE CASOS POSITIVOS Y DEFUNCIONES POR ENTIDAD. 80
FIGURA 94. GRÁFICA QUE MUESTRA LA TASA DE MORTALIDAD POR ENTIDAD. 81
FIGURA 95. GRÁFICA QUE MUESTRA LA TASA DE LETALIDAD POR ENTIDAD. 82

ÍNDICE DE TABLAS

TABLA 1. DIFERENCIAS ENTRE LAS BD SQL Y NOSQL 18
TABLA 2. SECTORES POR GRUPO PARA LAS NUEVAS MUESTRAS DE PACIENTES. 47
TABLA 3. GRUPOS POR ENTIDADES DE RESIDENCIA DE ACUERDO A LA REGIÓN GEOGRÁFICA A LA QUE PERTENECEN. 47
TABLA 4. CANTIDAD DE REGISTROS POR CADA UNA DE LAS TABLAS DE LA BD. 61
TABLA 5. CANTIDAD DE PACIENTES REGISTRADOS CON COVID-19 POR MES DE ENERO 2020 A MAYO 2021. 68
TABLA 6. EVALUACIÓN DEL TEST SPLIT. 75
TABLA 7. DETALLES DE PRECISIÓN POR CLASE. 76
TABLA 8. MATRIZ DE CONFUSIÓN RESULTANTE DE LA EVALUACIÓN DEL MODELO CON EL ALGORITMO J48. 76
TABLA 9. ENTIDADES CON EL TOTAL DE POBLACIÓN. 77
TABLA 10. CASOS POSITIVOS Y TOTAL DE DEFUNCIONES A CAUSA DE COVID-19 POR ENTIDAD. 77
TABLA 11. TASA DE MORTALIDAD CALCULADA POR ENTIDAD FEDERATIVA. 78
TABLA 12. TASA DE LETALIDAD CALCULADA POR ENTIDAD 79

Capítulo 1. Introducción

El aprendizaje automático es una rama en evolución de los algoritmos computacionales que están diseñados para emular la inteligencia humana aprendiendo del entorno circundante. Se les considera el caballo de batalla en la nueva era de los llamados big data. Las técnicas basadas en el aprendizaje automático se han aplicado con éxito en diversos campos que van desde el reconocimiento de patrones, visión por computadora, ingeniería de naves espaciales, finanzas, entretenimiento y biología computacional hasta aplicaciones biomédicas y médicas [1].

Desde mediados del siglo XIX, el área de estudio de la Inteligencia Artificial (IA) se ha enfocado en representar con modelos formales el conocimiento que el humano genera en las diferentes ramas de la ciencia. Con el Procesamiento del Lenguaje Natural (PLN) se generan técnicas que permiten integrar modelos computacionales con la lingüística aplicada. Uno de los enfoques que se explora con interés especial, es el estudio y análisis de la información disponible en Internet, la cual, al no ser estructurada, requiere técnicas de representación que permitan integrarla a fin de lograr la formalización y dar respuesta a preguntas que involucren lenguaje natural por parte del usuario como comenta Alemán C. Y. [2].

La Web ha ido evolucionando, desde ser un repositorio meramente estático compuesto de texto (Web tradicional) hasta contribuir al aprendizaje colaborativo mediante redes sociales, blogs y otras herramientas que permiten publicar contenido sin necesidad de tener conocimientos sobre diseño o programación Web. Aunque este cambio ha sido sustancial, se tiene el problema del exceso de información no utilizable. Una de las primeras propuestas para manejar este problema es la Web Semántica, donde se adopta una visión del contenido de la World Wide Web no solo como un repositorio, sino que trata de representar el contenido para que sea accesible a las máquinas y éstas sean capaces de interpretar la información ofrecida como menciona Rettinger A. et al. [3]. La Web Semántica y el Procesamiento de Lenguaje Natural son dos áreas de investigación en las que coinciden, entre otras ramas del conocimiento como lo son: la Ingeniería Informática, la Lingüística y la Documentación.

Es por ello que en esta tesis se realiza un análisis para poder acceder a los diferentes repositorios de información biomédica que se encuentran en la Web y obtener datos referentes a la pandemia de COVID-19 que se vive en el mundo, principalmente se trabaja con información proporcionada por la Secretaría de Salud en México, se lleva a cabo en 4 fases importantes que son: la búsqueda de información de pacientes con COVID-19, el pre-procesamiento de los datos obtenidos para obtener una estructura homógenea y poder trabajar con ellos, el procesamiento de los mismos para llevar a cabo el uso de herramientas de software que nos permitan el almacenamiento de grandes volúmenes de datos y finalmente realizar el análisis de patrones en los datos mediante técnicas de aprendizaje automático para proporcionar características descriptivas del conjunto de datos de aprendizaje y proveer algunas tendencias.

1.1 La epidemia de COVID-19

La información que proporciona Li. Q. [4] menciona que en diciembre 2019 surge un brote epidémico de neumonía de causa desconocida en Wuhan, provincia de Hubei de la República Popoular de China. Según el Centro Chino para el Control y Prevención de Enfermedades (CCDC), el 29 de diciembre un hospital en Wuhan admitió a 4 individuos con neumonía, quienes trabajaban en un mercado de esa ciudad. El hospital informó esto al CCDC (Centro Chino de Control y Prevención de Enfermedades),

cuyo equipo en la ciudad inició una investigación. El equipo encontró más casos relacionados al mercado y el 30 de diciembre las autoridades de salud de Wuhan comunicaron los casos al CCDC, que envió expertos a Wuhan para apoyar la investigación. Se obtuvieron muestras de estos pacientes para realizar análisis de laboratorio.

El 7 de enero de 2020 los científicos chinos habían aislado el virus causante de la enfermedad, y realizaron la secuenciación del genoma. Esta secuenciación estuvo disponible para la OMS el 12 de enero de 2020, permitiendo a los laboratorios de diferentes países producir diagnósticos específicos vía pruebas de PCR (Reacción en Cadena de la Polimerasa).

Así fue como en la última fecha mencionada, las autoridades chinas habían confirmado la existencia de 41 personas infectadas con el nuevo virus, quienes comenzaron a sentir síntomas entre el 8 de diciembre de 2019 y el 2 de enero de 2020, los cuales incluían: fiebre, malestar, tos seca, dificultad para respirar y fallos respiratorios; 33 también se observaron infiltrados neumónicos invasivos en ambos pulmones observables en las radiografías de tórax.

Por otra parte en [5] mencionan que los coronavirus son una extensa familia de virus que pueden causar enfermedades tanto en animales como en humanos. En los humanos, se sabe que varios coronavirus causan infecciones respiratorias que pueden ir desde el resfriado común hasta enfermedades más graves como el síndrome respiratorio de Oriente Medio (MERS) y el síndrome respiratorio agudo severo (SRAS). El coronavirus que se ha descubierto más recientemente causa la enfermedad de la cual se ha comentado conocida como COVID-19.

1.2 Problema de salud en México debido al COVID-19

El primer caso o caso índice de COVID-19 en México se detectó el 27 de febrero de 2020 en la Ciudad de México. Se trataba de un mexicano que había viajado a Italia y tenía síntomas leves. El 28 de febrero se confirmaron dos casos más: un italiano de 35 años, residente de la Ciudad de México, y un ciudadano mexicano del estado de Hidalgo que se encontraba en el estado de Sinaloa. Los dos habían viajado recientemente a Italia. La fase 1 de COVID-19 comenzó ese día. En esta fase, los casos de infección son importados del extranjero y no hay casos de contagio local; el número de personas infectadas con el virus es limitado y no hay medidas estrictas de salud, excepto acciones con el objetivo de difundir las acciones preventivas [6].

El 14 de marzo de 2020, la Secretaria de Educación Pública (SEP) adelantó el período de vacaciones de Semana Santa, extendiéndolo a un mes, del 23 de marzo al 20 de abril en todas las instituciones educativas de todo el país. El 18 de marzo se reportaron 118 casos confirmados de COVID-19, un aumento de 26% en comparación con el resultado del día anterior (93 casos). Ese mismo día, la Secretaría de Salud confirmó la primera muerte por COVID19 en México [6].

El gobierno federal decretó el 24 de marzo el inicio de la fase 2 de la pandemia COVID-19 en el país, tras registrar las primeras infecciones locales. En esta fase se suspenden principalmente ciertas actividades económicas, se restringen las congregaciones masivas y se recomienda permanecer en el domicilio a la población en general, especialmente a los mayores de 60 años y a las personas con diagnóstico de hipertensión arterial, diabetes, enfermedad cardíaca o pulmonar, inmunosupresión inducida o adquirida, a las mujeres que se encuentren en estado de embarazo o puerperio inmediato. A partir del 26 de marzo se suspendieron las actividades no esenciales del gobierno federal, exceptuando las relacionadas con los servicios de seguridad, salud, energía y limpieza. Se recomienda el estornudo de etiqueta, el lavado de manos constante y la desinfección continua de áreas de uso público. Las personas que tienen los síntomas y se han confirmado con COVID-19 tienen que usar

mascarillas faciales para evitar el contagio de otras personas. El personal de atención médica debe portar el equipo necesario de protección personal para evitar contagios al identificar a los pacientes en riesgo y al ser internados en las instalaciones médicas [6].

El 30 de marzo, se decretó una emergencia de salud nacional en México, dada la evolución de casos confirmados y las muertes por la enfermedad. Esto condujo al establecimiento de medidas adicionales para su prevención y control, como la suspensión inmediata de actividades no esenciales en todos los sectores económicos del país durante un mes, hasta el 30 de abril.
El 21 de abril del 2020 se dio por iniciada la fase 3 por COVID-19 en México, ya que se tenía evidencia de brotes activos y propagación en el territorio nacional con más de mil casos. Las medidas tomadas en esta fase fueron la suspensión de actividades no esenciales del sector público, privado y social, así como la extensión de la Jornada Nacional de Sana Distancia hasta el 30 de mayo [6].

Hoy en día al 26 de junio de 2021 se tiene un registro por parte de la Secretaría de Salud de México en conjunto con el CONACYT [7] de 2 503 408 casos confirmados, 4 503 137 casos negativos, 440 421 casos sospechosos, un total de 232 521 defunciones, 1 988 096 recuperados y 32 954 casos activos. Pero en realidad la cantidad de casos positivos estimados es de 2 689 334; en cuanto a defunciones estimadas se registran 244 945 y el número de activos estimados ronda los 35 833 casos (como se observa en la Figura 1).

Figura 1. Datos de COVID-19 en México

En cuanto a datos estadísticos dados en porcentajes, se tiene que los casos confirmados el 49.94% son mujeres y el 50.06% son hombres, además de que el 18.54% se encuentran hospitalizados y el 81.46% son ambulatorios (ver Figura 2).

Figura 2. Porcentajes de casos confirmados

Por otra parte en los datos de comorbilidades principales se reporta que de la misma manera en función al número de casos confirmados, el 16.87% sufren de hipertensión, el 14.01% de obesidad, el 12.97% de diabetes y el 7.26% de tabaquismos (ver Figura 3).

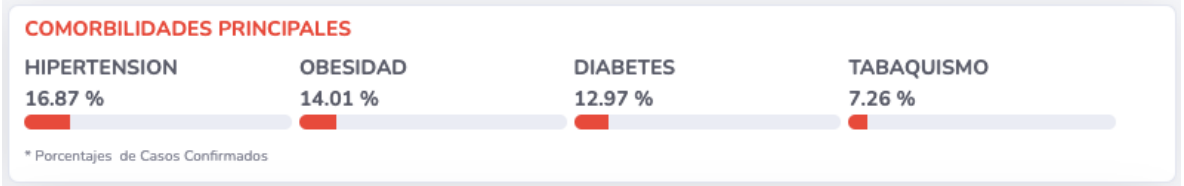


Figura 3. Porcentajes de acuerdo a comorbilidades principales

La siguiente gráfica proporcionada en [7] muestra el número de casos diarios clasificados por género y acumulados a nivel nacional (ver Figura 4).

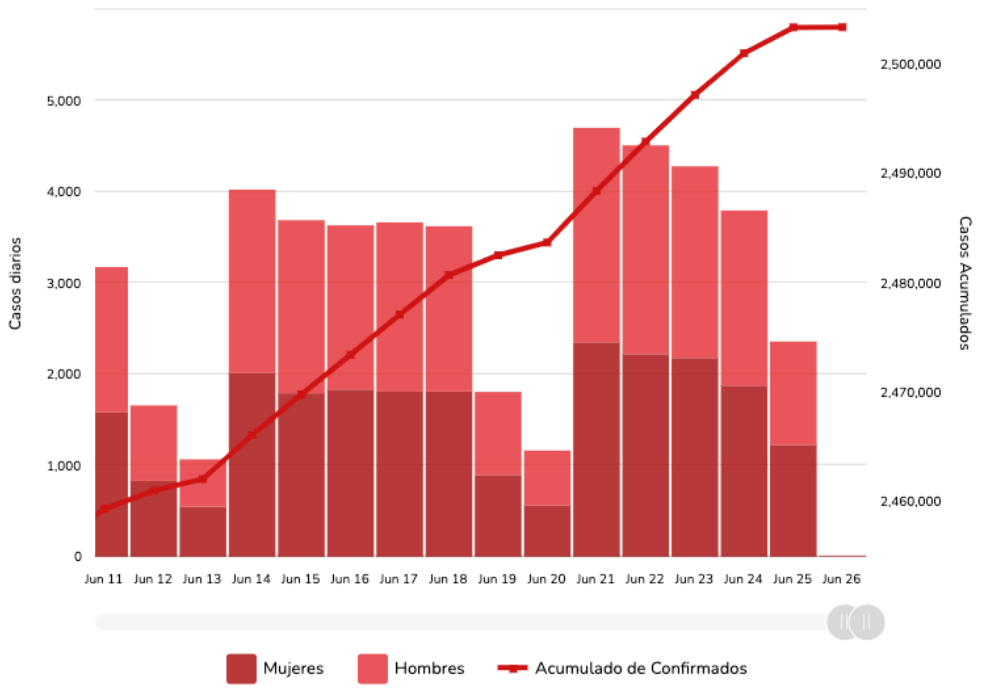


Figura 4. Gráfica de casos confirmados

Estadísticas de casos confirmados clasificados en rangos por edad y sexo (ver Figura 5).

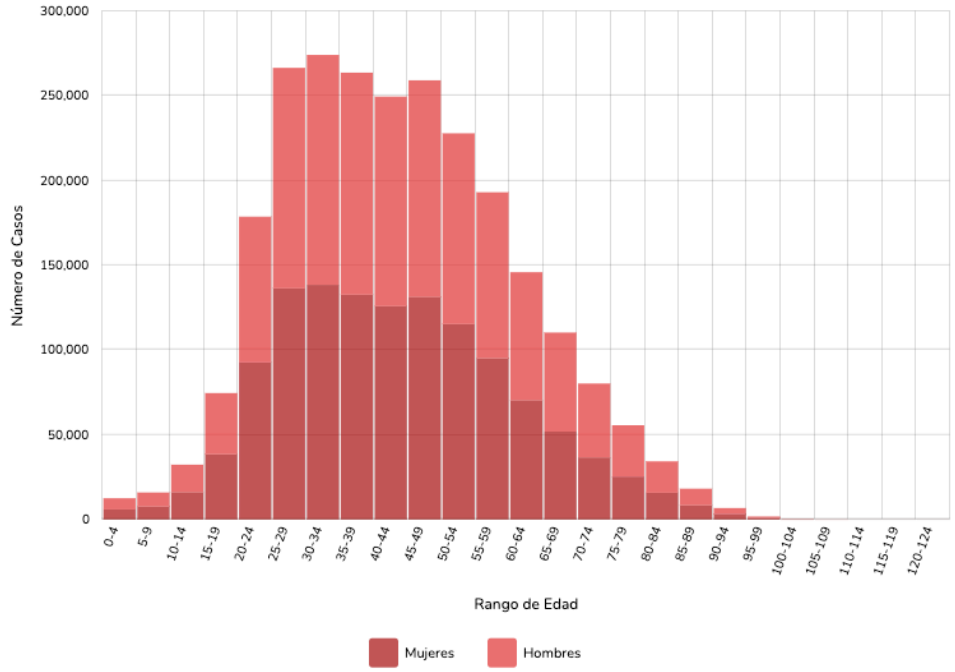


Figura 5. Histograma de casos confirmados a nivel nacional

Estadísticas de casos confirmados clasificados en rangos de edad y tipo de paciente (ver Figura 6).

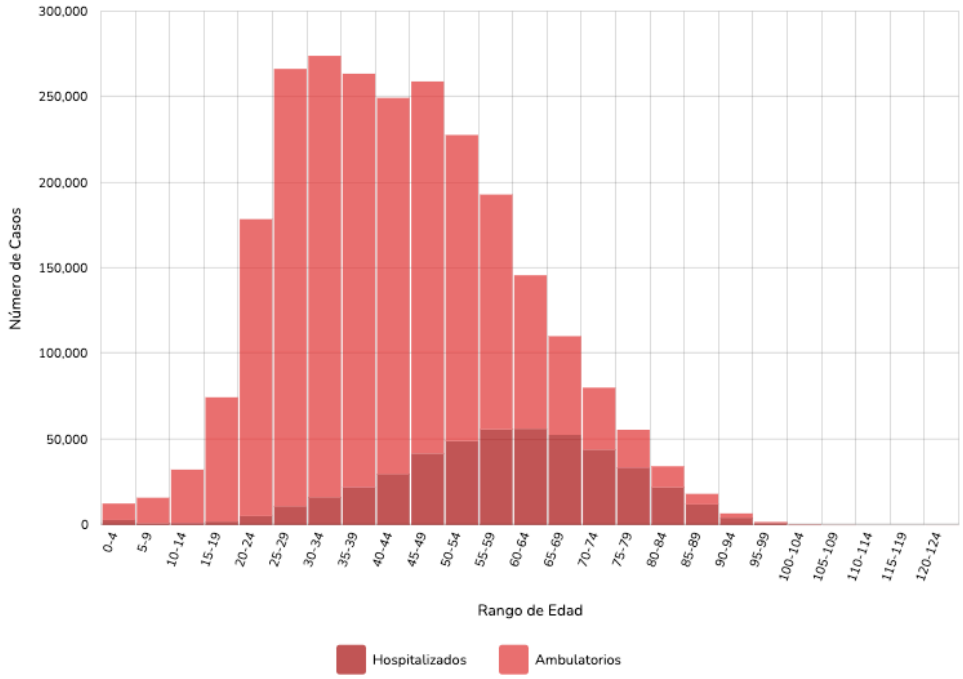


Figura 6. Histograma con rangos de edad y tipo paciente

1.2.1 Semáforo COVID-19

El Semáforo de riesgo epidemiológico para transitar hacia una nueva normalidad, es un sistema de monitoreo para la regulación del uso del espacio público de acuerdo con el riesgo de contagio de COVID-19.

Este semáforo será estatal y está compuesto por cuatro colores:

- **Rojo:** Se permitirán únicamente las actividades económicas esenciales, asimismo se permitirá también que las personas puedan salir a caminar alrededor de sus domicilios durante el día.

- **Naranja:** Además de las actividades económicas esenciales, se permitirá que las empresas de las actividades económicas no esenciales trabajen con el 30% del personal para su funcionamiento, siempre tomando en cuenta las medidas de cuidado máximo para las personas con mayor riesgo de presentar un cuadro grave de COVID-19, se abrirán los espacios públicos abiertos con un aforo (cantidad de personas) reducido.

- **Amarillo:** Todas las actividades laborales están permitidas, cuidando a las personas con mayor riesgo de presentar un cuadro grave de COVID-19. El espacio público abierto se abre de forma regular, y los espacios públicos cerrados se pueden abrir con aforo reducido. Como en otros colores del semáforo, estas actividades deben realizarse con medidas básicas de prevención y máximo cuidado a las personas con mayor riesgo de presentar un cuadro grave de COVID-19.

- **Verde:** Se permiten todas las actividades, incluidas las escolares.

El semáforo epidemiológico en México proporcionado en [8] al día 26 de junio de 2021 es el siguiente como se muestra en la Figura 7.

Figura 7. Semáforo epidemiológico por estado en México

En todos los niveles de riesgo epidemiológico se recomienda mantener las medidas de básicas de prevención que se muestran en la Figura 8.

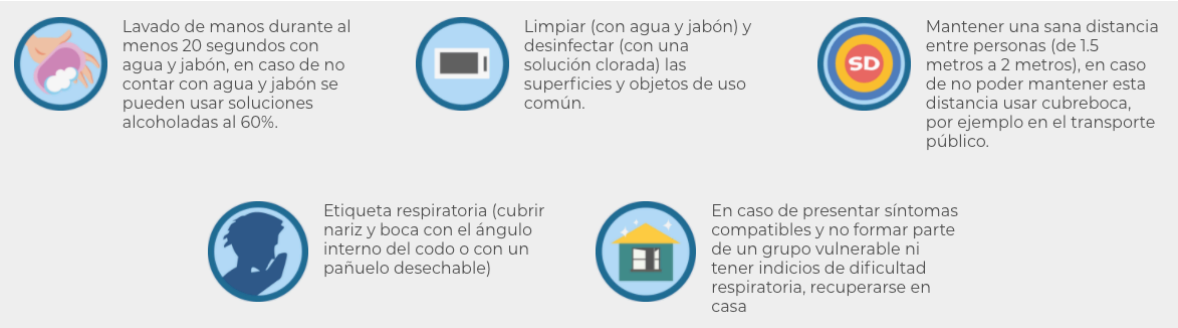


Figura 8. Medidas básicas de prevención

1.3 Definición del problema

La enfermedad que hoy en día está afectando a gran cantidad de la población a nivel mundial conocida como COVID-19, nombrada así por la Organización Mundial de la Salud (OMS), se propaga de manera exponencial en cada uno de los paises, tal es el caso de México, es por ello que el 11 de marzo de 2020 fue declarada pandemia por la OMS. Muchos estudios han reportado que esta enfermedad se puede transmitir de persona a persona a través del contacto directo con un individuo infectado o cuando una persona tose o estornuda y genera unas pequeñas gotitas que quedan suspendidas en el aire las cuales se pueden desplazar hasta 1.8 metros aproximadamente. También se ha detectado que el virus se puede transmitir a través del contacto con superficies contaminadas.

Dado que esta enfermedad es un problema de salud pública, es necesario proponer soluciones que permitan analizar los datos disponibles en internet acerca de la detección y tratamiento de los pacientes detectados con COVID-19. Mediante la información obtenida se pretenden analizar los diferentes datos que pueden ser las comorbilidades que presentan las personas infectadas como: neumonía, diabetes, asma, inmunosupresión, hipertensión, obesidad, tabaquismo, entre otras, puesto que en cada persona enferma se presentan de manera distinta, además de tener registros de las fechas en que presentaron los síntomas, fechas de ingreso en el caso que sean pacientes hospitalizados o bien defunciones.

El objetivo es llevar un control de toda la información, partiendo de su preprocesamiento, el cual nos permite obtener una estructura homogénea de los datos. Dicha información facilita el análisis para obtener patrones o bien puntos clave, que ayuden a obtener nuevos datos relevantes que complementen el análisis de la información obtenida y así, contribuir en el dominio de salud en México complementando los registros que se proporcionan respecto a esta pandemia en cuanto a los pacientes que han sido víctimas del COVID-19.

Se utilizarán herramientas como Python, el cual es un lenguaje de programación que provee librerías aptas que nos permiten llevar a cabo la parte del pre-procesamiento de la información para poder preparar los datos de acuerdo a las clasificaciones y modificaciones necesarias, además de usar otras herramientas de software que se mencionarán a lo largo del desarrollo de esta tesis.

Todos estos datos y herramientas nos serán útiles para llevar a cabo el procesamiento de la información, obtener características relevantes en los datos para implementar las bases de datos en los softwares correspondientes. Una vez concluido este proceso, se procede a la aplicación de técnicas de Big Data para la preparación de los datos y después utilizar técnicas de aprendizaje automático para llevar a cabo el análisis de patrones.

1.4 Justificación

Debido al ingente volumen de documentos disponibles en Internet, se hace imperativa la necesidad de utilizarlos para el desarrollo de proyectos de alto impacto a nivel nacional y global. El área de la salud pública demanda del aprendizaje a partir de los datos biomédicos disponibles en la Web. Estos datos se encuentran disponibles en formato heterogéneo y sin una estructura adecuada para ser procesados automáticamente.

Producto del brote del COVID-19 en China, se ha registrado una rápida propagación a escala internacional, con un aumento exponencial del número de casos y muertes, por lo cual la OMS declaró una emergencia de salud pública de alcance internacional. En consecuencia, la OMS está colaborando estrechamente con expertos mundiales y gobiernos para ampliar rápidamente los conocimientos científicos sobre este nuevo virus, rastrear su propagación, virulencia, asesorar a los países y a las personas sobre las medidas para proteger la salud y prevenir la propagación del virus.

Las herramientas de software derivadas de este proyecto de tesis, así como la obtención nuevos valores referentes a la pandemia por COVID-19 podrían ser utilizados para describir y/o predecir relaciones entre estos datos o anticipar valores de variables, las cuales aportarán conocimiento importante en situaciones críticas. Es por ello, que la creación de nuevas herramientas de software en el área de bases de datos constituye una tarea primordial en los proyectos que involucren aprendizaje automático. Por lo tanto, este proyecto podría ofrecer un método para la creación de nuevos repositorios de datos para tratar la información acerca de esta pandemia en México.

1.5 Objetivos y preguntas de investigación

A partir de esta investigación se planea generar las bases para una herramienta que pueda ser utilizada en el área de Ciencias de la Salud y la Computación para el manejo de información en relación al COVID-19.

El objetivo general de este trabajo de tesis puede establecerse como sigue:

Desarrollar una metodología de aprendizaje semiautomático de datos en el dominio de Biomedicina.

Para lograrlo, debe trabajarse en los siguientes objetivos específicos:

1. Evaluar las herramientas de software y los repositorios de información disponibles en la Web para el área de Ciencias de la Salud y así recopilar datos de pacientes que han padecido COVID-19.

2. Desarrollar una herramienta de software que permita el almacenamiento y procesamiento de la información recopilada.

3. Descubrir patrones o nuevas características en los datos para proporcionar información adicional referente a los pacientes con COVID-19.

Analizando los objetivos de la investigación se propone la siguiente pregunta en general: ¿Es posible desarrollar una metodología de aprendizaje semiautomática haciendo uso de gestores de bases de datos y nuevas tecnologías para el almacenamiento y procesamiento de la información en relación a la pandemia por COVID-19 para aportar nuevas herramientas de software y datos importantes referentes a los pacientes que sufren esta enfermedad?

En relación con los objetivos particulares, se trabajan las siguientes preguntas de investigación:

1. ¿Cuáles son los repositorios de información biomédica en la Web más adecuadas para recopilar información acerca del COVID-19?

2. ¿Qué lenguaje de programación es el más adecuado para llevar a cabo el pre-procesamiento y procesamiento de la información?

3. ¿Qué herramientas de software se pueden desarrollar para el almacenamiento, estructuración y procesamiento de los datos recopilados?

4. ¿Por qué utilizar un sistema gestor de bases de datos clásico y uno que presente la aplicación de nuevas tecnologías como lo es el almacenamiento en la nube para insertar registros de pacientes con COVID-19?

5. ¿Cuál es el conocimiento que puede generarse utilizando técnicas de aprendizaje automático para el análisis del conjunto de información recopilada?

1.4 Alcances y limitaciones

Considerando los objetivos que se proponen, se genera una metodología que pueda ser aplicada en el dominio de biomedicina en relación con la enfermedad que se ha presentado en todo el mundo que es el COVID-19, específicamente se utilizarán técnicas de recuperación de información, big data y aprendizaje automático. La investigación involucra las fases de la búsqueda de información de pacientes que han dado positivo a esta enfermedad; el pre-procesamiento de la información recopilada para obtener una estructura homogénea de los datos y llevar a cabo el diseño e implementación de una base de datos utilizando el DBMS MariaDB; el procesamiento una vez que se tiene la estructura requerida para que después se lleve a cabo la implementación en un gestor de bases de datos basado en la nube y finalmente el análisis de patrones en los datos extraídos mediante técnicas de minería de datos.

El conjunto de datos de entrenamiento estará conformado por información obtenida de artículos científicos o bien de repositorios que se encuentran en la Web que brindan información del dominio de la salud, principalmente datos proporcionados por la OMS y la Secretaría de Salud del Gobierno de México, por lo que tanto la creación y poblado de dicha base de datos estará sujeta al contenido de las fuentes de información mencionadas anteriormente.

Una característica importante de este conjunto de datos, es que los conceptos que se consideran principales tienen diferentes enfoques teóricos en el dominio, donde cada enfoque tiene una subclasificación distinta. Por lo tanto, cada corpus extraído para la obtención de información debe tener el mismo enfoque que los demás que integran el conjunto de datos de pacientes con COVID-19. Esta característica hace que en las fases de desarrollo de la metología, se tenga que actualizar la información de forma manual, debido a que los registros se actualizan día con día por parte de la Secretaría de Salud, para que después se pueda proceder al desarrollo utilizando las herramientas de software y posteriormente utilizar técnicas de aprendizaje automático que ayudarán a incrementar el número de instancias, así como también, deducir patrones que asistirán en el área de la Salud para proporcionar información nueva e importante referente a la pandemia causada por COVID-19.

Capítulo 2. Marco teórico

En este capítulo se muestran los conceptos teóricos y las herramientas de software necesarias para llevar a cabo el desarrollo de este trabajo de tesis.

2.1 Conceptos teóricos

A continuación se presentan los conceptos teóricos más importantes en relación al proceso que se debe seguir para desarrollar la metodología propuesta en este trabajo de tesis.

2.1.1 Recuperación de información

La recuperación de información como se define en [9] es el conjunto de actividades orientadas a facilitar la localización de determinados datos u objetos, y las interrelaciones que estos tienen a su vez con otros. Existen varias disciplinas vinculadas a esta actividad como la lingüística, la documentación o la informática.

Tradicionalmente se limitaba a la recuperación de documentos escritos, el término se redefinió para incorporar la creciente aparición de materiales multimedia. Así, los nuevos buscadores de información en Internet, que originariamente buscaban textos, expandieron su actividad a imágenes, videos o audios. De esta forma términos como recuperación de textos, recuperación documental y recuperación de información son utilizados como equivalentes.
Por otro lado, la necesidad de localizar datos concretos ha ido expandiendo su área de aplicación. En la actualidad se está migrando desde la recuperación de documentos a la recuperación pregunta-respuesta, que responden con el dato concreto y no con el conjunto de documentos que posiblemente contenga este dato [9].

Con todo este tema que se ha mencionado, es como surgen los sistemas de recuperación de información, ya que el desarrollo de estas aplicaciones informáticas surgió como respuesta a la gestión de la sobreabundancia de información actual. La forma en que esta información es almacenada suele ser mediante Bases de Datos y repositorios documentales.

El funcionamiento de un sistema de recuperación de información se puede medir analizando los datos (o documentos) recuperados ante una consulta [9]. Dos son las principales medidas:

- Precision: volumen de datos relevantes entre el total de datos recuperados.
- Exhaustividad: volumen de datos relevantes entre el total de datos relevantes en el repositorio o la BD.

Ambas medidas tienden a evolucionar en sentido inverso (Ley de Cleverdon). Cuanto más crece la precisión más disminuye la exhaustividad, y al contrario. Esto es debido a que miden factores distintos, el ruido y el silencio:

- Ruido: información recuperada no relevante.
- Silencio: información no recuperada que es relevante.

Dado que para calcular estas medidas es necesario conocer cuantos elementos relevantes existen, son necesarios listados de la relevancia de los documentos ante un conjunto de consultas. Estos listados

se llaman colecciones de pruebas (test collections), y son utilizadas en competiciones internacionales para testear los sistemas de recuperación [9].

Lenguajes de indización y control terminológico

Índices

Se definen en [10] como un listado de términos normalizados que representan el contenido de un recurso. Algunos tipos son:

- Índice de materias: Términos ordenados según las materias que trata la base de datos, el buscador, etc.
- Índice alfabético: Listado de términos alfabéticamente.
- Índice KWIC: Tipo de índice permutado en el que el contenido temático de una obra se representa mediante palabras clave de su título o de otra fuente de información del documento.
- Índice KWOC: Tipo de índice permutado que varía en su presentación respecto al índice KWIC, en que las palabras clave aparecen como un encabezamiento en línea separada. Bajo cada encabezamiento aparece la totalidad de los títulos, completos o truncados, que contienen la palabra clave de que se trata.

Palabras clave (Keywords)

Término significativo en lenguaje natural [10] que representa el contenido del documento. En la búsqueda de información esta opción es esencial ya que nos permite acotar y precisar información. El problema recae en definir la palabra exacta que representa el contenido, por ello es conveniente utilizar especificadores.

Tesauros

Como se define en [10], es un listado terminológico controlado sobre un área o ámbito de conocimiento que mantiene entre sí relaciones semánticas y genéricas. Su principal característica es que los términos están ordenados jerárquicamente, permitiendo la precisión terminológica en la búsqueda de información.

Componentes:

- Descriptores admitidos o preferentes: son aquellos términos normalizados (donde han sufrido un proceso de expurgo denegando plurales, evitando sinónimos, etc.) que el tesauro los considera aptos para asignarlos a un documento y que posteriormente facilite la recuperación.
- Descriptores no admitidos: son aquellos que aun estando normalizados no se consideran adecuado para utilizarlos (suelen ser sinónimos, términos no utilizados en el campo de actuación, etc.).

Relaciones:

- Jerárquicas: Indican cuando un término es más específico que otro.
- Asociativas: Indican que los términos guardan alguna relación.
- Sinónimos: Indican que dos términos son sinónimos y cual de ellos se utiliza como admitido.

2.1.2 Ontologías

La palabra Ontología se deriva del griego ontos (estudio del ser) y logos (palabra). Filosóficamente, es la ciencia de qué es, es una explicación sistemática de la existencia, de los tipos de estructuras, categorías de objetos, propiedades, eventos, procesos y relaciones en cada área de la realidad como lo define Smith B. [13]. En las aplicaciones de la vida real, una ontología es una entidad computacional, y no ha de ser considerada como una entidad natural que se descubre, sino como un recurso artificial que se crea (Mahesh K. [14]).

Mientras que en las ciencias computaciones Gruber T. [15] define una ontología como una especificación formal de una conceptualización. Otra definición que proporciona Weigand H. [16] es que una ontología se define como una base de datos que describe los conceptos en el mundo o algún dominio, algunas de sus propiedades y cómo los conceptos se relacionan entre sí. Esta base de datos se define a partir de un corpus base, del cual se extraen elementos principales o palabras clave. Posteriormente, del mismo texto se infieren las relaciones entre palabras clave, de esta manera, se crea una estructura de grafo donde los nodos son las palabras clave y las aristas representan la relación que existe entre ellas [2].

Entre las aplicaciones más representativas de las ontologías se encuentran la representación formal del conocimiento, lo que facilita el manejo e integración de datos con estructuras diferentes. Formalmente, una ontología se define como la sextupla O = (C, H, I, R, P, A) Faria C. et al. [17] donde:

- C - es el conjunto de entidades de la ontología.
- H - son las relaciones taxonómicas entre los conceptos.
- I - indica las relaciones entre instancias.
- R - es el conjunto de relaciones no taxonómicas.
- P - es el conjunto de propiedades de la ontología.
- A - representa el conjunto de axiomas y reglas que prueban la consistencia de la ontología y que realizan el proceso de inferencia.

Heijst V. & Guarino N. [18] clasifican las ontologías de acuerdo a dos dimensiones ortogonales: la cantidad y el tipo de estructura de la conceptualización y el tema de la conceptualización. En la primera dimensión, se distinguen tres categorías: las ontologías terminológicas como léxicos, ontologías de información como base de datos de esquemas y modelado de conocimiento y las ontologías que especifican las conceptualizaciones de los conocimientos. En la segunda dimensión, se identifican cuatro categorías:

- Ontología de aplicación: Ontología de procesos de producción, de diagnóstico de fallas, de diseño intermedio de barcos.
- Ontología de dominio: Son específicas para un tipo de artefacto, generalizaciones sobre tareas específicas en algún dominio concreto del conocimiento. Por ejemplo: ontología del proceso de producción.
- Ontología de técnicas básicas: Describen características generales de artefactos, por ejemplo componentes, procesos y funciones.
- Ontologías genéricas: Describe la categoría de más alto nivel, describiendo conceptos generales (tiempo, espacio, objeto).

En una ontología, los conceptos representan la base para la descripción de la información. Esta descripción se realiza mediante tres componentes: Términos, atributos y relaciones. Los términos son

nombres utilizados para referirse a un concepto específico que puede incluir un conjunto de sinónimos que especifican los mismos conceptos. Los atributos describen el concepto a detalle utilizando características, y las relaciones se utilizan para representar correspondencias entre diferentes conceptos y proveer una estructura general de la ontología, como lo menciona Sánchez S. [19].

2.1.3 Big Data

Debido a la gran cantidad de datos que se encuentran hoy en día dependiendo su dominio, es como surgen nuevas necesidades para poder procesar y almacenar la información. Big data se define en [34] como datos que contienen una mayor variedad y que se presentan en volúmenes crecientes y a una velocidad superior. Esto se conoce como "las 3V" (volumen, velocidad y variedad de datos), aunque en los últimos años, han surgido otras "dos V": valor y veracidad.

Dicho de otro modo en [34], el big data está formado por conjuntos de datos de mayor tamaño y más complejos, especialmente procedentes de nuevas fuentes de datos. Estos conjuntos de datos son tan voluminosos que el software de procesamiento de datos convencional sencillamente no puede administrarlos. Sin embargo, estos volúmenes masivos de datos pueden utilizarse para abordar problemas que antes no hubiera sido posible solucionar.

Las 3V de Big data que se definen en [35] son:

Volumen: La cantidad de datos importa. Hace referencia a la cantidad de datos que se generan y recopilan constantemente. Con big data, se tendrán que procesar grandes volúmenes de datos no estructurados de baja densidad, pueden tratarse de datos de valor desconocido.

Velocidad: Continuamente se están generando datos, por lo que las plataformas de Big Data deben ser capaces de acceder a ellos, almacenarlos y tratarlos para que las organizaciones puedan tomar mejores decisiones basadas en la información que aportan. Por lo tanto, la velocidad se refiere a la rapidez con que los datos son creados, accedidos, almacenados y procesados.

Variedad: Los datos no son homogéneos, es decir, no tienen una única característica común a todos ellos que permita procesarlos a todos de la misma manera y en una única iteración. Los datos son muy diversos ya que provienen de distintos orígenes o fuentes, son de distintos tipos y tienen distintos formatos. Además, pueden ser estructurados, semiestructurados o no estructurados. Todas estas particularidades es lo que se conoce como variedad de los datos y es una de las características fundamentales que deben tener los sistemas de procesamiento de Big Data: deben ser capaces de procesar datos de diversas formas, tipos y fuentes.

Ventajas de big data y del análisis de datos proporcionadas en [34]:

- El big data le permite obtener respuestas más completas, ya que dispone de mayor cantidad de información.
- La disponibilidad de respuestas más completas significa una mayor fiabilidad de los datos, lo que implica un enfoque completamente distinto a la hora de abordar problemas.

La ciencia de datos es un proceso iterativo donde su ciclo de vida está compuesto por seis etapas principales como la planificación, construcción de un modelo de datos, evaluación del modelo, explicación del modelo, implementación y monitorización del modelo [36].

2.1.4 Aprendizaje Automático

El aprendizaje automático se define en [37] como una aplicación de inteligencia artificial (IA) que proporciona a los sistemas la capacidad de aprender y mejorar automáticamente a partir de la experiencia sin estar programados explícitamente. El aprendizaje automático se centra en el desarrollo de programas informáticos que pueden acceder a los datos y utilizarlos para aprender por sí mismos.

El proceso de aprendizaje comienza con observaciones o datos, como ejemplos, experiencia directa o instrucción, con el fin de buscar patrones en los datos y tomar mejores decisiones en el futuro basados en los ejemplos que proporcionamos. El objetivo principal es permitir que las computadoras aprendan automáticamente sin intervención o asistencia humana y ajustar las acciones en consecuencia. Pero, utilizando los algoritmos clásicos del aprendizaje automático, el texto se considera una secuencia de palabras clave; en cambio, un enfoque basado en el análisis semántico imita la capacidad humana para comprender el significado de un texto [37].

En ciencia de datos [38], un algoritmo es una secuencia de pasos de procesamiento estadístico. En el aprendizaje automático, los algoritmos están 'entrenados' para encontrar patrones y características en cantidades masivas de datos con el fin de tomar decisiones y predicciones basadas en datos nuevos. Cuanto mejor sea el algoritmo, más precisas serán las decisiones y predicciones a medida que procesa más datos.

Los métodos conocidos de aprendizaje automático presentados en [38] son:

- Aprendizaje automático supervisado
- Aprendizaje automático no supervisado
- Aprendizaje semi-supervisado

Existen 4 pasos básicos presentados en [38] para crear una aplicación o modelo de aprendizaje automático:

1. **Seleccionar y preparar el conjunto de datos de entrenamiento:** Los datos de entrenamiento son un conjunto de datos representativos de los datos que el modelo de aprendizaje automático ingiere para resolver el problema para el que está diseñado.

2. **Elegir un algoritmo para ejecutar en el conjunto de datos de entrenamiento:** Un algoritmo es un conjunto de pasos de procesamiento estadístico. El tipo de algoritmo depende del tipo (etiquetado o no etiquetado) y la cantidad de datos en el conjunto de datos de entrenamiento y del tipo de problema a resolver.

3. **Entrenar el algoritmo para crear el modelo:** Entrenar el algoritmo es un proceso iterativo: implica ejecutar variables a través del algoritmo, comparar la salida con los resultados que debería haber producido, ajustar pesos y sesgos dentro del algoritmo que podrían producir un resultado más preciso y ejecutar las variables nuevamente hasta que el algoritmo devuelve el resultado correcto la mayor parte del tiempo. El algoritmo entrenado y preciso resultante es el modelo de aprendizaje automático.

4. **Usar y mejorar el modelo:** El último paso es utilizar el modelo con nuevos datos y, en el mejor de los casos, mejorar su precisión y eficacia con el tiempo. El origen de los nuevos datos dependerá del problema que se resuelva.

2.2 Web Scraping

Mitchel R. [22] define el Web Scraping como la práctica de recopilar datos a través de cualquier otro medio que un programa que interactúa con una API (o, obviamente, a través de un humano que utiliza un navegador web). Esto se logra más comúnmente escribiendo un programa automatizado que consulta un servidor web, solicita datos (generalmente en forma de HTML y otros archivos que componen las páginas web) y luego analiza esos datos para extraer la información necesaria.

En la práctica, el web scraping abarca una amplia variedad de técnicas de programación y tecnologías como análisis de datos, análisis de lenguaje natural e información de seguridad. Los crawlers de banda son excelentes para recolectar y procesar grandes cantidades de datos rápidamente. En general, la extracción de datos web es utilizada por personas y empresas que desean hacer uso de la gran cantidad de datos web disponibles públicamente para tomar decisiones más inteligentes.

Por otra parte Hiremath O. S. [23] lo define como un método automatizado que se utiliza para extraer grandes cantidades de datos de sitios web. Los datos de los sitios web no están estructurados. El Web Scraping ayuda a recopilar estos datos no estructurados y a almacenarlos de forma estructurada. Hay diferentes formas de extraer sitios web, como servicios en línea, API o escribir su propio código.

Los fundamentos del Web Scraping

Es muy simple y funciona mediante dos partes como se menciona en [24]: un web crawler y un web scraper. El web crawler es el caballo y el scraper es el carro. El crawler conduce el scraper, como de la mano, a través de Internet, donde extrae los datos solicitados.

- **Crawler:** Un web crawler, que generalmente le llaman “araña”, es una inteligencia artificial que navega por Internet para indexar y buscar contenido siguiendo enlaces y explorando, como una persona con demasiado tiempo libre. En muchos proyectos, primero "rastrea" la web o un sitio web específico para descubrir las URL que luego pasa a su scraper.

- **Scraper:** Un web scraper es una herramienta especializada diseñada para extraer datos de una página web de forma precisa y rápida. Los web scrapers varían ampliamente en diseño y complejidad, según el proyecto. Una parte importante de cada scraper son los localizadores de datos (o selectores) que se utilizan para encontrar los datos que desea extraer del archivo HTML; normalmente se aplica xpath, selectores css, regex o una combinación de ellos.

Proceso de Web Scraping:

1. Identificar el sitio web de destino.
2. Recopile las URL de las páginas de las que desea extraer datos.
3. Realice una solicitud a estas URL para obtener el HTML de la página.
4. Use localizadores para encontrar los datos en el HTML.
5. Guarde los datos en un archivo JSON o CSV o en algún otro formato estructurado.

2.3 Python

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con tipado dinámico y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripts o pegamento para conectar componentes existentes. La sintaxis simple y

fácil de aprender de Python enfatiza la legibilidad y por lo tanto reduce el costo de mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en formato fuente o binario sin cargo para todas las plataformas principales y se pueden distribuir libremente [25].

Las ventajas del lenguaje Python que se proporcionan en [26] son las siguientes:

Simplificado y rápido: Este lenguaje simplifica mucho la programación "hace que te adaptes a un modo de lenguaje de programación, Python te propone un patrón". Es un gran lenguaje para scripting, si se requiere algo rápido (en el sentido de la ejecución del lenguaje), con unas cuantas líneas ya está resuelto.

Elegante y flexible: El lenguaje le da muchas herramientas, si se requieren listas de varios tipo de datos, no hace falta que declares cada tipo de datos. Es un lenguaje tan flexible no se deben preocupar tanto por los detalles.

Programación sana y productiva: Programar en Python se convierte en un estilo muy sano de programar: es sencillo de aprender, direccionado a las reglas perfectas, le hace como dependiente de mejorar, cumplir las reglas, el uso de las lineas, de variables". Ademas es un lenguaje que fue hecho con productividad en mente, es decir, Python hace ser mas productivo, permite entregar en los tiempos que se requieren.

Ordenado y limpio: El orden que mantiene Python, es de lo que más le gusta a sus usuarios, es muy legible, cualquier otro programador lo puede leer y trabajar sobre el programa escrito en Python. Los módulos están bien organizados, a diferencia de otros lenguajes.

Portable: Es un lenguaje muy portable (ya sea en Mac, Linux o Windows) en comparación con otros lenguajes. La filosofía de baterías incluidas, son las librerías que se necesitan al día a día de programación, ya están dentro del intérprete, no tienen la necesidad de instalarlas adicionalmente como en otros lenguajes.

Comunidad: Algo muy importante para el desarrollo de un lenguaje es la comunidad, la misma comunidad de Python cuida el lenguaje y casi todas las actualizaciones se hacen de manera democrática.

En Hiremath O. S. [23] menciona una lista de caracterísicas por las cuales Python es más adecuado para llevar a cabo Web Scraping:

- **Facilidad de uso:** Python es fácil de codificar. No es necesario agregar punto y coma ";" o llaves "{}" en cualquier lugar. Esto lo hace menos complicado y fácil de usar.
- **Gran colección de bibliotecas:** Python tiene una gran colección de bibliotecas como Numpy , Matlplotlib , Pandas , etc., que proporcionan métodos y servicios para diversos fines. Por lo tanto, es adecuado para Web Scraping y para una mayor manipulación de los datos extraídos.
- **Tipeado dinámicamente:** en Python, no se tienen que definir tipos de datos para las variables, se pueden usar directamente las variables donde sea necesario. Esto ahorra tiempo y acelera su trabajo.
- **Sintaxis fácilmente comprensible:** principalmente porque leer un código Python es muy similar a leer una declaración en inglés. Es expresivo y fácil de leer, y la sangría utilizada en

Python también ayuda al usuario a diferenciar entre diferentes alcances / bloques en el código.

- **Código pequeño, tarea grande:** el Web Scraping se utiliza para ahorrar tiempo, por eso en Python, se pueden escribir códigos pequeños para realizar tareas grandes. Por lo tanto, ahorra tiempo incluso mientras se escribe el código.

De la misma manera, Hiremath O. S. [23] comenta que, cuando ejecuta el código para Web Scraping, se envía una solicitud a la URL que ha mencionado. Como respuesta a la solicitud, el servidor envía los datos y le permite leer la página HTML o XML. Luego, el código analiza la página HTML o XML, encuentra los datos y los extrae.
Para extraer datos utilizando Web Scraping con Python, se deben seguir estos pasos básicos: Buscar la URL que desea raspar (scrape), inspeccionar la página, encontrar los datos que se desean extraer, escribir el código, ejecutar el código y extraer los datos, almacenar los datos en el formato requerido.

2.4 BeautifulSoup

Beautiful Soup en [27] se define como una biblioteca de Python para obtener datos de HTML, XML y otros lenguajes de marcado. Supongamos que ha encontrado algunas páginas web que muestran datos relevantes para su investigación, como información de fecha o dirección, pero que no proporcionan ninguna forma de descargar los datos directamente. Beautiful Soup lo ayuda a extraer contenido particular de una página web, eliminar el marcado HTML y guardar la información. Es una herramienta para raspado web que le ayuda a limpiar y analizar los documentos que ha extraído de la web.

Características de BeautifulSoup

Algunas características clave mencionadas en [28] que hacen única esta herramienta son:

- Beautiful Soup proporciona algunos métodos simples y modismos Pythonic para navegar, buscar y modificar un árbol de análisis.
- Beautiful Soup convierte automáticamente los documentos entrantes a Unicode y los documentos salientes a UTF-8.
- Beautiful Soup se encuentra en la parte superior de los analizadores de Python populares como lxmly html5lib, lo que nos permite probar diferentes estrategias de análisis o cambiar la velocidad por flexibilidad.

2.5 Gestores de Datos

Un Sistema de Gestión de Bases de Datos (SGBD o DBMS) se define en [29] como un software que proporciona una forma de almacenar y recuperar la información de una base de datos de manera práctica y eficiente. Ofrece una interfaz entre la base de datos y los usuarios finales o aplicaciones, asegurando que los datos estén organizados de manera consistente y que sean fácilmente accesibles. De esta manera, los usuarios pueden crear, leer, actualizar y eliminar datos de una BD.

SQL vs NoSQL

En [29] se definen las Bases de datos como conjuntos de información organizada para ser usadas o consultadas posteriormente. Estas se han vuelto una necesidad para cualquier software. Se utilizan a diario de forma directa o indirecta.

Existen en la actualidad diversos tipos de bases de datos. Estos están diseñados para atender características y necesidades especiales. Para navegar en el océano digital se debe conocer la distinción fundamental entre las bases de datos relacionales y no relacionales.

También se presentan algunas ventajas y desventajas expuestas en la información proporcionada en [15], como lo son:

Ventajas de las bases de datos SQL:

- **Experiencia y madurez:** En el caso para las BD SQL este es uno de sus puntos más fuertes. El tiempo y la aceptación generalizada de los desarrolladores ha permitido crear gran cantidad de información y herramientas en torno a ellas.
- **Atomicidad:** Los desarrolladores generalmente se ven dispuestos a inclinarse por los modelos relacionales gracias a la atomicidad. Esto significa que cualquier operación que se quiera ejecutar y no cumpla con los criterios de información preestablecidos, no se realizará.

- **Estándares bien definidos:** Todos los procesos deben estar bajo los estándares que plantea el SQL. Brindando de esta forma criterios de uniformidad a la información.

- **Escritura simple:** Gran parte de la aceptación depende de la sencillez de su método de escritura. Este es muy parecido al lenguaje que utilizamos los humanos, facilitando para nosotros la comprensión de las operaciones.

Ventajas de las bases de datos NoSQL:

Versatilidad: La capacidad de adaptación para brindar soluciones es el punto más fuerte de las NoSQL. Las posibilidades de crecimiento en el volumen de datos o la posibilidad de incluir cambios sobre la forma en la que se ingresan los datos sin necesidad de alterar la estructura, permite adaptarse de forma rápida a un entorno de alto dinamismo como el mundo de hoy.

Crecimiento horizontal: Estas bases de datos son altamente escalables. Es decir, que si se requiere instalar mayor cantidad de nodos para ampliar la capacidad, se puede hacer sin problemas. Esto no va a interrumpir la usabilidad o consultas dentro de la BD.

Bajos requerimientos: No se necesitan servidores con gran cantidad de recursos para operar. La adaptabilidad y flexibilidad en las BD NoSQL permiten empezar con bajos niveles de inversión en equipos e ir ampliando la capacidad a medida de las necesidades que surgen.

Algunas diferencias importantes entre las bases de datos SQL y NoSQL que se mencionan en [16] son las siguientes (ver Tabla 1):

Tabla 1. Diferencias entre las BD SQL y NoSQL

BD SQL	BD NoSQL
Son relacionales.	Son no relacionales.
Utilizan un lenguaje de consulta estructurado y tienen un esquema predefinido.	Tienen esquemas dinámicos para datos no estructurados.
Son escalables verticalmente.	Son escalables horizontalmente.
Se basan en tablas.	Son almacenes de documentos, valores clave, gráficos o columnas anchas.

Son mejores para transacciones de varias filas.	Son mejores para datos no estructurados como documentos o JSON.

2.5.1 ElasticSearch

En el trabajo presentado por Miñambres G. et al. [20] se define como un servidor de búsqueda basado en Lucene. Provee de un motor de búsqueda de texto completo, distribuido y con capacidad de dar servicio a múltiples clientes a través de una interfaz web RESTful y utilizando como entrada archivos JSON. Está desarrollado en Java y actualmente está publicado como código abierto bajo las condiciones de la licencia Apache.

Esta herramienta nos permite indexar un gran volumen de datos para posteriormente hacer consultas sobre ellos soportando entre muchas otras cosas búsquedas aproximadas, facetas y resaltado. Al estar todos los datos indexados, los resultados se obtienen de formar muy rápida. Lo único que hay que hacer es añadir los ficheros JSON con sus propiedades y ElasticSearch se encarga de indexarlo y asignarle un identificador para que más tarde la búsqueda se ejecute rápidamente.

Por otra parte en [30] se menciona que es utilizado para el análisis de registros, búsqueda de texto completo, inteligencia de seguridad, análisis de negocios y casos de uso de inteligencia operacional. Accesible a través de una API extensa y elaborada, Elasticsearch puede impulsar búsquedas extremadamente rápidas que admiten sus aplicaciones de descubrimiento de datos.

Características principales:

- Operaciones de búsqueda casi en tiempo real.
- Alto rendimiento.
- Facilita la coubicación de datos.
- Indización de documentos al repositorio.
- Almacenamiento de documentos sin formato.
- Altamente distribuible y escalable.

Como complemento, en [31] se dice que trabaja en conjunto con Kibana, herramienta la cual le permite explorar, visualizar y compartir información sobre sus datos de forma interactiva para así poder administrar y monitorear la pila.

Elasticsearch proporciona búsquedas y análisis casi en tiempo real para todo tipo de datos. Ya sea que tenga texto estructurado o no estructurado, datos numéricos o datos geoespaciales, esta herramienta puede almacenarlos e indexarlos de manera eficiente de tal forma que admita búsquedas rápidas. Se recalca que puede ir mucho más allá de la simple recuperación de datos y agregar información para descubrir tendencias y patrones en sus datos, a medida que crece su volumen de datos y consultas, la naturaleza distribuida de Elasticsearch permite que su implementación crezca sin problemas junto con ella.

2.5.2 MariaDB

Rouse M. [32] define MariaDB como un sistema de administración de bases de datos relacionales (DBMS) de código abierto que es un reemplazo directo compatible para la tecnología de base de datos MySQL ampliamente utilizada. Fue creado como una bifurcación de software de MySQL por desarrolladores que desempeñaron un papel clave en la construcción de la base de datos original; idearon MariaDB en 2009 en respuesta a la adquisición de MySQL por Oracle Corp. Se basa en SQL y admite el procesamiento de datos al estilo ACID con atomicidad, consistencia, aislamiento

y durabilidad garantizados para las transacciones. Admite múltiples motores de almacenamiento, incluidos InnoDB, MyRocks, Spider, Aria, TokuDB, Cassandra y MariaDB ColumnStore.

Por otra parte en [33] se hace enfásis en que MariaDB convierte los datos en información estructurada en una amplia gama de aplicaciones, que van desde la banca hasta los sitios web. Originalmente diseñado como un reemplazo directo mejorado para MySQL, MariaDB se utiliza porque es rápido, escalable y robusto, con un rico ecosistema de motores de almacenamiento, complementos y muchas otras herramientas que lo hacen muy versátil para una amplia variedad de casos de uso.

Algunas de las características más importantes presentadas en [30] acerca de MariaDB son las siguientes:

Admite API de JSON:

- Replicación de datos en paralelo.
- Múltiples motores de almacenamiento.
- Notablemente escalable.
- Amplia selección de motores de almacenamiento.
- Utiliza un lenguaje de consulta estándar y popular.
- Velocidad y alta seguridad.

Capítulo 3. Estado del arte

En esta sección se presenta un análisis de los diferentes trabajos desarrollados en el área de aprendizaje automático en los dominios de salud referentes en el caso clínico de COVID-19. Para esto se analizan los artículos relacionados con la creación, uso de herramientas de software y dominios utilizados en esta investigación.

3.1 COVID-19

Dentro del proceso de revisión de la literatura, se encontraron autores como Mena R. H. et al. [39] los cuales presentan el análisis del modelo epidemiológico que se puede utilizar como una herramienta de pronóstico para hacer predicciones y evaluar planes de contingencia. Se usan estadísticas Bayesianas para incorporar observaciones y usar la gama completa de estimaciones de parámetros contenidas en la distribución posterior para ajustar las incertidumbres en las predicciones del modelo. Se utiliza una distribución de probabilidad para las observaciones para generar todas las curvas epidemiológicas posibles a través de la distribución predictiva posterior. Este enfoque se aplica para evaluar si los planes de contigencia en CDMX están teniendo éxito y si la curva se ha aplanado.

Por otra parte, Miñambres F. et al. [20] presentan en este trabajo el desarrollo de una herramienta la cual ayudará al médico a encontrar el diagnóstico de un paciente en el menor tiempo posible, dicho proyecto tiene como objetivo ayudar al personal médico ofreciéndoles la posibilidad de comparar un gran número de informes en busca de aquellos más similares respecto a la información que se extrae de cada informe analizado. Es por ello que en el desarrollo se presentan tres fases importantes. el preprocesamiento de los historiales médicos obtenidos, el procesamiento en cuanto a secciones médicas para una representación estructurada y la búsqueda de similitud para encontrar aquellos historiales más similares en base a los conceptos médicos extraídos. Se utilizan distintas herramientas importantes, una de ellas conocida como UMLS, la cual maneja gran cantidad de términos médicos que ayudan en el procesamiento de los informes, así como ElasticSearch, el cual es un motor de búsqueda que permitirá poder llevar a cabo el proceso de búsqueda de similitud y así conseguir el software que se desea implementar.

Jeon J. et al. [40] en este artículo presentan la investigación y análisis de literatura biomédica y datos de las redes sociales públicas para comprender la asociación de los factores de riesgo y los síntomas con varios resultados de los pacientes con COVID-19. Se llevó a cabo el análisis semántico, recopilando 45 estudios de cohorte retrospectivos y 84140 publicaciones en Twitter de 1036 usuarios positivos para COVID-19. Se utilizaron herramientas de aprendizaje automático para extraer información biomédica para identificar síntomas poco comunes o novedosos que se mencionan en las redes sociales, para después examinar y comparar dos conjuntos de datos para expandir el panorama de factores de riesgo y síntomas relacionados con COVID-19. Por lo que con el resultado de este análisis obtuvieron 72 factores de riesgo que se asociaron específicamente con los resultados individuales. Se compilan estudios retrospectivos que investigan variables clínicas y demográficas de varios resultados de COVID-19. Se generan dos conjuntos de datos de palabras claves, el primer conjunto representado basado en cohortes y estudios retrospectivos; el segundo conjunto representa COVID-19.

Mientras tanto, Wang S. et al. [41] proponen un sistema deep learning completamente automático para el análisis de diagnóstico y pronóstico de COVID-19 mediante tomografía computarizada de uso rutinario, se recolectaron retrospectivamente 5372 pacientes con imágenes de tomografía

computarizada de siete ciudades o provincias de China. Para entrenar previamente el sistema deep learning, se utilizaron 4106 pacientes con imágenes de tomografía computarizada, haciendo que el sistema aprenda las características pulmonares; posteriormente se inscribieron 924 pacientes con COVID-19 y 342 con neumonía para capacitar y validar externamente el desempeño del sistema. El uso de deep learning proporciona una herramienta conveniente para la detección rápida de COVID-19 e identificación de posibles pacientes de alto riesgo, lo que puede ser útil para la optimización de los recursos médicos y la prevención temprana antes de que los pacientes muestren síntomas graves.

En este ámbito enfocado con la pandemia causada por COVID-19, Barrientos-Gutiérrez T. et al. [42] presentan ideas que necesitan un desarrollo urgente y colaborativo. Se discute la estimación de la magnitud de la epidemia mediante un panel nacional de seroprevalencia y nuevas estrategias para mejorar el monitoreo en tiempo real de la epidemia. También se analizan las externalidades negativas asociadas con la respuesta a la pandemia. De la misma manera, se presenta un marco general para el desarrollo de ideas para salir del confinamiento, resaltando la importancia de implementar acciones estructurales, sostenibles y equitativas. Se hace un llamado a la solidaridad y la cooperación, donde esfuerzos y creatividad se dediquen a la resolución de los problemas que enfrentan México y el mundo.

En cuanto al trabajo presentado por Wu G., Yang P. et al. [43] pretenden desarrollar y validar un modelo de machine learning basado en características clínicas para la evaluación de riesgo de gravedad y el triaje (selección) de pacientes con COVID-19 al ingreso hospitalario. Para poder entrenar y validar este modelo se utilizaron 725 pacientes, incluyendo un grupo de 299 pacientes de Wuhan, China, así como otros 426 pacientes de ocho centros en China, Italia y Bélgica. El resultado principal fue la aparición de una enfermedad grave o crítica durante la hospitalización. Los rendimientos del modelo se cuantificaron utilizando el método de área bajo la curva de características operativas del receptor (AUC) y métricas derivadas de la matriz de confusión. Es por ello que hacen mención que este modelo machine learning, el nanograma y las calculadoras en línea (página específica: https://covid19risk.ai) podrían ser útiles para acceder a la aparición de enfermedades graves y críticas entre los pacientes con COVID-19 y la clasificación en el momento de ingreso hospitalario.

Torrealba-Rodríguez O. et al. [44] en este trabajo presentan el modelado y la predicción de casos de infección por COVID-19 en México a través de modelos matemáticos y computacionales utilizando solo los casos confirmados proporcionados por el informe técnico diario COVID-19 MEXICO hasta el 8 de mayo. Se aplicaron los modelos matemáticos: Gompertz y Logístico, así como el modelo computacional: Red Neural Artificial para realizar la modelación del número de casos de infección.

Haciendo mención en el mismo ámbito de la enfermedad conocida como COVID-19 estudiando análisis clínicos, Hurtado-Duarte A. et al. [45] en este artículo presentan un análisis para trabajar en cuanto a los hallazgos tomógraficos en la afectación pulmonar por COVID-19, se utiliza la tomografía computarizada, ya que es fundamental en el proceso de diagnóstico dada su alta sensiblidad, junto con ello, se implementa un estudio transversal analítico a pacientes atendidos en el Instituto Nacional de Enfermedades Respiratorias, Ciudad de México, con definición operacional vigente de caso sospechoso para COVID-19, que contaran con prueba de RT-PCR específica y tomografía computarizada de tórax en la valoración inicial. Se realizó un análisis descriptivo y analítico mediante la prueba de $\chi 2$ y t de Student. Se empleó el programa Epi-Info versión 7. Se analizaron 56 pacientes, con una edad promedio de 51 años, 61% fue del sexo masculino. 52% presentaron comorbilidades, siendo la diabetes mellitus la más frecuente. Los síntomas que principalmente se observaron fueron fiebre, tos y cefalea. El patrón tomográfico que predominó fue mixto, con localización subpleural y bilateral.

Enfermedades crónicas como la obesidad y la diabetes, son principales factores los cuales provocan complicaciones mayores en pacientes que se han enfermado por COVID-19; es por ello que, Bello-Chavolla, O. Y. et al. [48] presentan un análisis indicando los factores de riesgo y proponen una puntuación clínica para predecir la letalidad de la enfermedad causa de esta pandemia, incluidos los factores específicos para la diabetes y la obesidad, así como su papel en la mejora de la predicción del riesgo.

Para ello, se obtuvieron datos de casos de COVID-19 confirmados y negativos así como sus características demográficas y de salud de la Dirección General de Epidemiología de la Secretaría de Salud de México. Se investigaron factores de riesgo específicos asociados con la positividad y la mortalidad del virus explorando el impacto de la diabetes y la obesidad en la modificación de la letalidad relacionada con la enfermedad. Finalmente, se construye una puntuación clínica para predecir la letalidad de COVID-19.

Los factores de riesgo de letalidad en COVID-19 incluyen diabetes de aparición temprana, obesidad, enfermedad pulmonar obstructiva crónica, edad avanzada, hipertensión, inmunosupresión y enfermedad renal crónica (ERC); se obtuvo que la obesidad media el 49.5% del efecto de la diabetes sobre la letalidad del virus. La diabetes de inicio temprano confiere un mayor riesgo de hospitalización y la obesidad confiere un mayor riesgo de ingreso en la unidad de cuidados intensivos e intubación.

Debido a que algunas comorbilidades están asociadas con la enfermedad grave por coronavirus (Covid-19) pero no está claro si algunas aumentan la susceptibilidad a Covid-19, Hernández-Garduño, E. [49] presenta un estudio mexicano de casos y controles, encontrando que la obesidad representa el predictor más fuerte de Covid-19 seguido de diabetes e hipertensión en ambos sexos e insuficiencia renal crónica solo en mujeres. El tabaquismo activo se asoció con una disminución de las probabilidades de contraer Covid-19. Esos hallazgos indican que estas comorbilidades no solo están asociadas con la gravedad de la enfermedad, sino que también predisponen a contraer Covid-19. Se necesitan investigaciones futuras para establecer los mecanismos involucrados en cada comorbilidad y el aparente efecto "protector" del tabaquismo.
Este estudio utilizó la base de datos Covid-19 disponible públicamente de la Secretaría de Salud de México a través del sitio web de la Dirección General de Epidemiología a partir del cual se obtuvo información de todos los pacientes evaluados para Covid-19 al 15 de mayo de 2020. Las variables en la base de datos incluyen identificación no nominal (asignada al azar), edad, sexo, fumador actual, historial de contacto con esta enfermedad, tipo de paciente: ambulatorio contra hospitalizado y si el paciente estaba o no ingresado en la unidad de cuidados intensivos (UCI) o había sido intubado (intubación traqueal para ventilación mecánica). Finalmente obteniendo un análisis univariable demostrando los principales factores que se asocian a los síntomas y a los pacientes que han padecido esta enfermedad.

Adicionalmente a las literaturas presentadas anteriormente, se presenta el estudio de Parra-Bracamonte, G. M. et al. [50] cuyo propósito fue evaluar las características clínicas y factores de riesgo de mortalidad de pacientes con enfermedad por coronavirus (COVID-19) de México, dado que se encuentra en transmisión comunitaria activa. En dicha investigación, se ajustaron al modelo de regresión logística multivariante y a las curvas de supervivencia de Kaplan-Meier para estudiar las probabilidades de muerte de las características y comorbilidades en pacientes con COVID-19 en México. Algunos resultados significativos que proporciona esta investigación son que la edad, el sexo y las comorbilidades más frecuentes diabetes, obesidad e hipertensión se asociaron significativamente con el riesgo de muerte por coronavirus, mientras que el hábito de fumar no se identificó como un factor de riesgo mortal pero por otra parte, comorbilidades menos frecuentes

como la enfermedad pulmonar obstructiva crónica, la enfermedad renal crónica y los pacientes con enfermedades inmunodeprimidas también mostraron un riesgo significativo de muerte.

3.2 Aplicaciones de la ontología SNOMED-CT y CIDO Ontology

En el trabajo desarrollado por Miñambres F. et al. [20] la ontología SNOMED-CT es utilizada para la parte del procesado de los informes médicos, es decir, anteriormente en el pre-procesado se obtuvieron las representaciones correspondientes en formato xml, ahora el objetivo del procesado es que se analicen dichas representaciones con el fin de identificar conceptos de la ontología, en otras palabras, se requiere procesar los documentos buscando conceptos respecto a SNOMED-CT. Esto es útil para poder trabajar con los diferentes términos médicos con los que ya cuenta dicha ontología.

Por otra parte Jani B. D. et al. [46] en su trabajo pretenden recopilar una lista de síntomas relevantes, elementos de evaluación, datos demográficos, condiciones de estilo de vida y de salud asociados con COVID-19, además de hacer coincidir estos elementos de datos con los términos clínicos de la ontología SNOMED-CT correspondientes para respaldar el desarrollo y la implementación de plantillas de consulta. Se utilizaron pautas y revisiones publicadas y no publicadas (N=61) para recopilar una lista de elementos de datos relevantes para las consultas de COVID-19. Se utilizó el navegador NHS Digital SNOMED CT para identificar identificadores descriptivos y de conceptos. Los desafíos clave de implementación se conceptualizaron a través de una lente de teoría del proceso de normalización (NPT). Es por ellos que se necesitan plantillas de consulta para COVID-19 para estandarizar la recopilación de datos, facilitar la investigación y el aprendizaje, y potencialmente mejorar la calidad de la atención para COVID-19.

El desarrollo de CIDO Ontology es relevante para aportación en el área de la salud debido a la problemática que se vive a causa de la pandemia por COVID-19. En el estudio coordinado por Y. He, H. Yu et al. [21] se requiere la integración de datos grandes y que crecen exponencialmente investigando sobre COVID-19 para comprender mejor su etiología, transmisión y mecanismo de patogénesis. Además, se debe ser capaz de traducir esa comprensión en un rápido desarrollo de la estratificación de pacientes.
Por otra parte, se menciona que es una ontología biomédica de código abierto en el área de enfermedades infecciosas por coronavirus. CIDO proporciona anotaciones y representaciones estandarizadas interpretables por humanos y computadoras. La ontología se utilizará como base de conocimientos de última generación para la representación estándar y lógica de conocimientos heterogéneos sobre coronavirus.

3.3 Aplicaciones en Procesamiento de Lenguaje Natural

En el trabajo presentado por Miñambres F. et al. [20] se tiene la utilización de WordNet para la herramienta de apoyo basada en el análisis de historias clínicas presentada en Miñambres F. et al. [2], en el que se extraen los conceptos necesarios en la parte del procesado de los informes clínicos, esto para ser clasificados por secciones médicas y a cada concepto se le asocia una etiqueta gramatical extraída del diccionario propio de FreeLing, así como el identificador de concepto extraído de la base de datos de WordNet.

Por otra parte, Huang K. et al. [47] proponen una metodología de sustitución de sinónimos de cuatro etapas utilizando WordNet. Se realiza un grupo de experimentos en los que se varían los dos parámetros metodológicos "número máximo de sustituciones por trimestre" y "duración máxima del plazo". El propósito es examinar su efecto sobre el crecimiento del número de sinónimos potenciales generados y la pérdida de resultados asociada. Los experimentos se basan en la reintegración de la

"Terminología mínima estándar" (MST) en UMLS. Las coincidencias de sustitución de sinónimos que no coinciden con el contenido actual del UMLS y, por lo tanto, se consideran incorrectas, se examinan más a fondo manualmente como una auditoría de la integración original del MST. Con el desarrollo de esta metodología de sustitución de sinónimos que utiliza WordNet es una ayuda automatizada útil en la integración de fuentes UMLS; también ayudó a descubrir errores en la integración original del MST y mejorar la calidad del contenido conceptual UMLS.

3.4 Aplicaciones en Machine Learning

Muhammad, L. J. et al . [51] presentan un trabajo en el cual desarrollaron modelos de aprendizaje automático supervisado para la infección por COVID-19 con algoritmos de aprendizaje que incluyen regresión logística, árboles de decisión, máquina de soporte vectorial, Bayes ingenuo y red artificial neutral utilizando un conjunto de datos etiquetado epidemiológico para casos positivos y negativos de COVID-19 en México. El análisis del coeficiente de correlación entre varias características dependientes e independientes se llevó a cabo para determinar una relación de fuerza entre cada característica dependiente y característica independiente del conjunto de datos antes de desarrollar los modelos. El 80% del conjunto de datos de entrenamiento se utilizó para entrenar los modelos, mientras que el 20% restante se utilizó para probar los modelos. Estas alternativas facilitarían el diagnóstico y pronóstico de los pacientes que han sido víctimas de la enfermedad que se vive actualmente que es el COVID-19.

En otro trabajo que presenta Hung M. et al. [52] se encargan de analizar las discusiones en Twitter relacionadas con COVID-19 e investigar los sentimientos hacia esta enfermedad. Este estudio aplicó métodos de aprendizaje automático en el campo de la inteligencia artificial para analizar los datos recopilados de Twitter. Utilizando tweets originados exclusivamente en los Estados Unidos y escritos en inglés durante el período de 1 mes del 20 de marzo al 19 de abril de 2020, el estudio examinó las discusiones relacionadas con COVID-19. También se realizaron análisis de redes sociales y sentimientos para determinar la red social de los temas dominantes y si los tweets expresaban sentimientos positivos, neutrales o negativos. También se realizó un análisis geográfico de los tweets. El estudio identificó 5 temas dominantes entre los tweets relacionados con COVID-19: entorno de atención médica, apoyo emocional, economía empresarial, cambio social y estrés psicológico. Alaska, Wyoming, Nuevo México, Pensilvania y Florida fueron los estados que expresaron el sentimiento más negativo, mientras que Vermont, Dakota del Norte, Utah, Colorado, Tennessee y Carolina del Norte transmitieron el sentimiento más positivo.

Finalmente en el trabajo presentado por Hernández-Galdamez, D. et al. [53] mencionan que la población de México tiene altas tasas de prevalencia de enfermades no transmisibles (ENT). La hospitalización y muerte de pacientes con COVID-19 en los países más afectados por la pandemia se ha asociado a comorbilidades crónicas. Es por eso que con ese estudio los autores prentenden describir la prevalencia de ENT en pacientes con COVID-19 en México y analizar el aumento de riesgo por comorbilidades y factores de riesgo sobre hospitalización, utilización de unidades de cuidados intensivos y muerte. Realizaron un estudio transversal a partir de 212,802 casos confirmados de COVID-19 notificados por el Ministerio de Salud hasta el 27 de junio de 2020. Las razones de probabilidad se realizaron mediante un modelo de regresión logística. Mediante esta investigación se obtiene que, las comorbilidades de las ENT aumentan la gravedad de la infección por COVID-19. Dadas las altas tasas de prevalencia de ENT entre la población mexicana, la pandemia representa una amenaza especial para el sistema de salud y la sociedad, es por ello que, mencionan la necesidad de fortalecer las medidas especiales de prevención para las personas con diagnóstico de ENT a corto plazo.

3.5 Análisis y limitantes

Es importante mencionar que muchos de los trabajos analizados anteriormente no proporcionan una metodología como tal enfocada en el área de biomedicina con el claso clínico de COVID-19. Mayormente estos trabajos se basan en el análisis de información proporcionada por redes sociales; o bien se enfocan en el desarrollo de nuevas herramientas tecnológicas con deep learning, analizando el impacto de las enfermedades crónicas en conjunto con el COVID-19. En este proyecto de tesis, se propone una metodología de aprendizaje semiautomática de datos para el dominio de biomedicina, comenzando desde la adquisición de datos de fuentes de internet, la representación de los mismos hasta el aprendizaje automático.

La aportación fundamental de este trabajo consiste en el desarrollo de una metodología, la cual se basa en la recopilación de datos de pacientes con COVID-19 en México, así como procesar toda la información recabada dándole una estructura homogénea a los datos para que después sea posible hacer uso de nuevas tecnologías. Estas nuevas tecnologías comprenden, un motor de búsqueda basado en la nube llamado ElasticSearch, el cual permite almacenar grandes volúmenes de información, pero a su vez combinando esta implementación con el uso de MariaDB como gestor de base de datos para tener herramientas propias de SQL, por lo que el uso de estas dos herramientas garantiza una herramienta de software propia para el análisis y almacenamiento de la información referente a los pacientes con COVID-19. Además, se lleva a cabo aprendizaje automático mediante la aplicación de técnicas de minería de datos en conjunción con programación en Python, que permite llevar a cabo los cálculos necesarios para poder brindar datos como son el índice de mortalidad que se obtiene de los pacientes que sufren esta enfermedad.

Capítulo 4. Metodología

En este capítulo se describe la metología propuesta para esta investigación, la cual inicialmente se divide en cuatro fases. En la primera fase, se lleva a cabo el proceso de la búsqueda de datos en repositorios de información biomédica establecidos, por lo que, se realiza toda la obtención de los registros de COVID-19 a utilizar; después en la segunda fase se lleva a cabo el pre-procesamiento de la información, es decir, darle la estructura adecuada a los datos para utilizarlos conforme nuestra necesidad; la tercera fase consta del procesamiento de la información, en la cual llevamos a cabo la creación de las bases de datos correspondientes para almacenar toda la información de los pacientes que se han contagiado de COVID 19 y finalmente en la cuarta fase se lleva a cabo el análisis de patrones en los datos mediante técnicas de minería de datos para después lograr obtener los resultados experimentales. El diagrama de la metodología se muestra en la Figura 9.

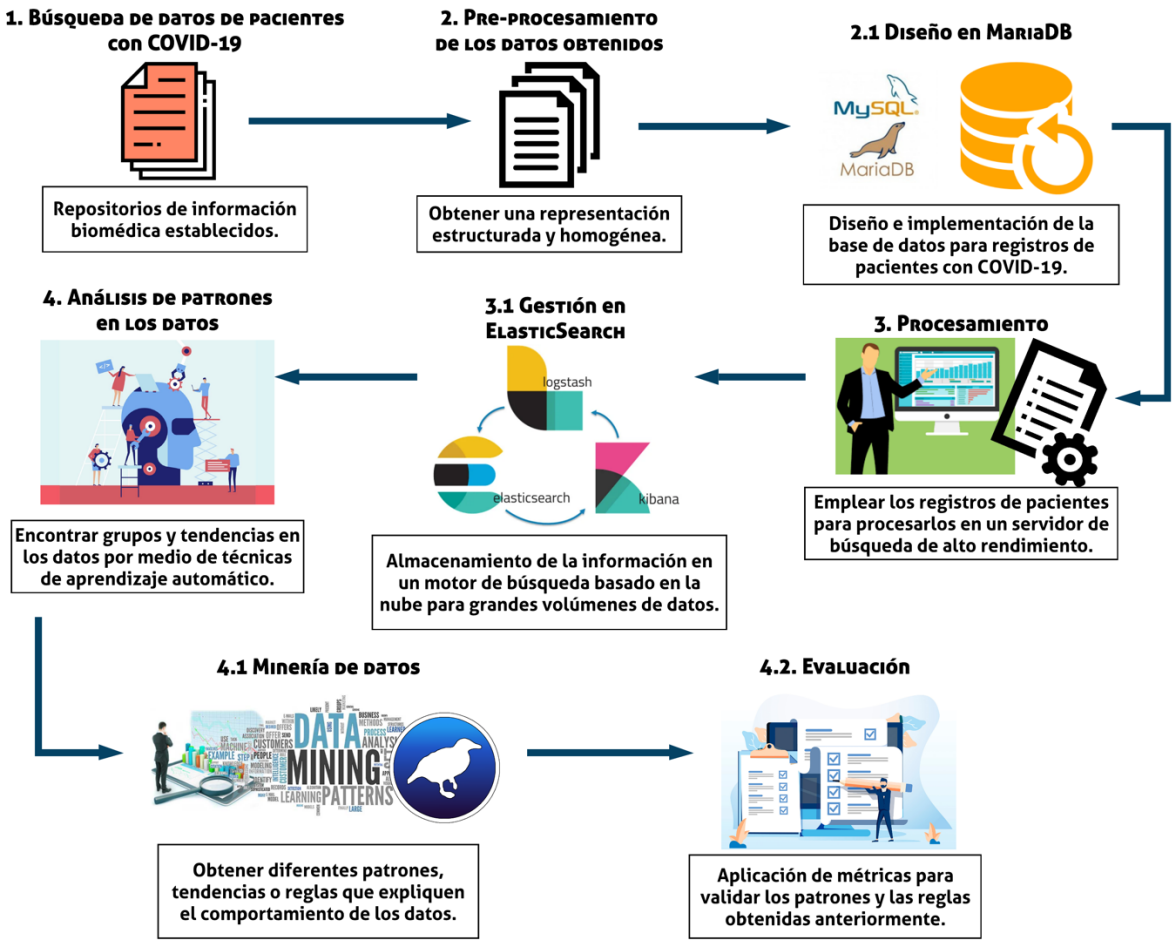


Figura 9. Metología propuesta para el desarrollo de este proyecto.

4.1 Fase 1: Búsqueda de datos COVID-19

En esta primera fase, se describe como se ha llevado a cabo la recopilación de datos de COVID-19, en la sección 4.1.1 el uso de Web Scraping para obtención de artículos científicos referentes a esta pandemia y en la 4.1.2 la descripción de la principal fuente de obtención de los datos para este proyecto de tesis.

Los recopilación de datos se ha llevado a cabo de diferentes maneras, ya que se han buscado fuentes de libre acceso para poder extraer atributos importantes y generalizados dentro de una estructura ordenada.

Inicialmente comenzamos la búsqueda de información en fuentes confiables en el ámbito de la salud, como lo son la Organización Mundial de la Salud, el Banco Mundial, la Organización Panamericana de la Salud, PubMed y la Secretaría de Salud en México junto con el CONACYT.

Figura 10. Fuentes de libre acceso para búsqueda de información COVID-19.

4.1.1 Web Scraping para búsqueda de documentos médicos

Se propone un método para búsqueda de información automatizada, el cual es el uso de scrappers, se programa un script mediante el lenguaje de programación Python, en el cual se define la estructura de los datos que se quieren recuperar de un sitio Web, esto nos ayuda a poder recolectar diferentes artículos científicos los cuales tratan información acerca del COVID-19.

Por ejemplo, se presenta un script el cual nos devuelve el título de un artículo científico encontrado en el sitio PubMed, así como también obtendrá el DOI de cada uno de ellos para respaldar la información y después tener acceso a los links donde se encuentran (ver Figura 11).

```python
import requests
from urllib.request import urlopen
from bs4 import BeautifulSoup
class Content:
    def __init__(self, url, title):
        self.url = url
        self.title = title

def getPage(url):
    req = requests.get(url)
    return BeautifulSoup(req.text, 'html.parser')


def scrapeBrookings(url):
    bs = getPage(url)
    for title,doi in zip(bs.find_all('a',{'class':'docsum-title'}),bs.find_all('span',{'class':'docsum-journal-citation full-journal-citation'})):
        temp=doi.get_text().split("doi:")[1]
        temp=temp.split(" ")[1]
        datos = Content(temp, title.get_text())
        print('Title: {}'.format(datos.title))
        print('DOI: https://doi.org/{}\n'.format(datos.url))
        input()
        pass

url = 'https://pubmed.ncbi.nlm.nih.gov/?term=covid+19'
content = scrapeBrookings(url)
```

Figura 11. Web scrapper programado en Python para obtener artículos científicos.

4.1.2 Principal fuente de obtención de datos COVID-19

Los datos que se han recopilado para esta tesis son provenientes del repositorio creado por el Gobierno de México en conjunto con el CONACYT, ya que posee una gran variedad de atributos como son: el id del paciente, la entidad y municipio donde residen, el tipo de paciente, la fecha de síntomas por COVID-19, fecha de defunción en los casos donde así haya ocurrido, datos específicos de diferentes enfermedades crónicas o si padecen algunas comorbilidades, entre muchos otros.

Esta información que se brinda en esta fuente de datos es de gran ayuda, ya que son una gran cantidad de pacientes registrados, desde inicios de la pandemia en México, de los cuales se muestra a detalle la información referente a los atributos mencionados anteriormente; estos registros los podemos encontrar ordenados por mes desde abril de 2020 hasta la fecha.

En la pagína principal de esta fuente de información encontramos por ejemplo, los diferentes registros hasta Junio de 2021 de los casos positivos registrados en el país, así como las defunciones estimadas, los casos activos estimados, casos sospechosos, el número de personas recuperadas, etc. Al igual que diferentes gráficas que nos brindan información estadística referente a la pandemia por la cual estamos pasando a nivel nacional. Es importante recalcar que este repositorio de datos es muy funcional, ya que se actualiza día con día (ver Figura 12).

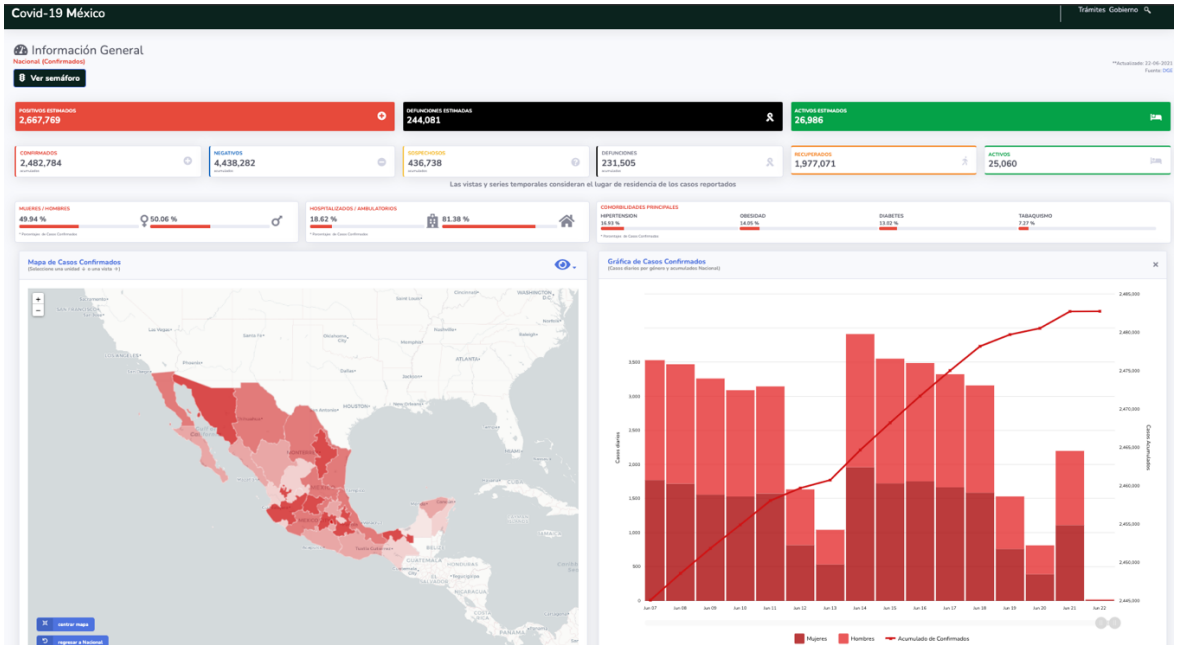


Figura 12. Tablero de información COVID-19 en la página del Gobierno de México.

Los datos que se han recabado para este trabajo de tesis se encuentran en formato separado por comas (CSV), los cuales tienen una estructura no homogénea que se tendrán que modificar para poder trabajar con ellos, pero lo importante a recalcar es que este tipo de formato permite trabajar de una manera adecuada con Python para poder llevar a cabo modificaciones en la información de acuerdo a nuestras necesidades.

Se cuenta con un archivo con 10 mil registros de pacientes para realizar pruebas y después implementar la base de datos, así como también el archivo con todos los registros completos que cuenta con más de 1 millón de pacientes que han padecido COVID-19 (ver Figura 13).

	FECHA_ACTU	ID_REGISTRO	ORIGEN	SECTOR	ENTIDAD_UM	SEXO	ENTIDAD_NA	ENTIDAD_RE	MUNICIPIO_I	TIPO_PACIEN	FECHA_INGR	FECHA_SINT	FECHA_DEF	INTUBADO	NEUMONIA	EDAD	NACIONALID	EMBARAZO
2	30/11/20	71735	2	9	21	2	21	21	114	1	18/03/20	12/03/20	9999-99-99	97	2	75	1	97
3	30/11/20	1c4583	2	12	9	2	9	9	4	1	30/03/20	30/03/20	9999-99-99	97	2	23	1	97
4	30/11/20	0d55c9	2	12	9	1	9	9	16	1	26/03/20	24/03/20	9999-99-99	97	2	28	1	2
5	30/11/20	1468a5	1	4	5	1	15	5	18	1	02/04/20	27/03/20	9999-99-99	97	2	47	1	2
6	30/11/20	043f64	2	4	9	2	9	9	10	2	26/03/20	26/03/20	30/03/20	2	1	58	1	97
7	30/11/20	0e07d8	1	4	15	2	15	15	104	2	28/03/20	28/03/20	02/04/20	2	1	49	1	97
8	30/11/20	13757c	1	12	15	1	15	15	106	1	31/03/20	27/03/20	9999-99-99	97	2	27	1	2
9	30/11/20	2371	1	4	3	2	3	3	8	1	31/03/20	27/03/20	9999-99-99	97	2	38	1	97
10	30/11/20	11fb00	1	12	9	2	9	9	15	1	29/03/20	25/03/20	05/04/20	97	1	54	1	97
11	30/11/20	92521	1	12	9	1	9	9	17	1	30/03/20	28/03/20	9999-99-99	97	2	49	1	2
12	30/11/20	0955a5	2	12	6	1	14	6	10	1	01/04/20	28/03/20	9999-99-99	97	2	38	1	2
13	30/11/20	1a1f12	2	4	14	2	9	14	67	2	27/03/20	27/03/20	31/03/20	2	1	42	1	97
14	30/11/20	12d93f	1	4	23	2	23	23	5	1	24/03/20	20/03/20	9999-99-99	97	1	46	1	97
15	30/11/20	197355	1	9	9	2	17	9	12	1	13/03/20	02/03/20	9999-99-99	97	2	27	1	97
16	30/11/20	0b3ad1	2	12	14	1	14	14	120	1	18/03/20	15/03/20	9999-99-99	97	2	39	1	2
17	30/11/20	03f951	2	12	11	1	11	11	20	1	25/03/20	20/03/20	9999-99-99	97	2	61	1	2
18	30/11/20	185d7d	1	12	9	1	9	9	10	1	11/03/20	11/03/20	9999-99-99	97	2	63	1	2
19	30/11/20	0e7ba2	1	12	11	2	11	11	7	1	27/03/20	26/03/20	9999-99-99	97	2	16	1	97
20	30/11/20	15ce6f	2	12	26	1	26	26	55	2	28/03/20	22/03/20	9999-99-99	2	1	47	1	2
21	30/11/20	0e7853	1	13	21	1	21	21	85	2	28/03/20	23/03/20	05/04/20	2	1	73	1	2
22	30/11/20	08de8a	1	12	9	2	9	15	57	1	28/03/20	27/03/20	9999-99-99	97	2	42	1	97
23	30/11/20	016eda	2	12	3	1	14	3	8	1	30/03/20	23/03/20	9999-99-99	97	2	29	1	2
24	30/11/20	0a9217	1	4	9	2	9	9	5	2	26/03/20	26/03/20	02/04/20	2	2	94	1	97
25	30/11/20	1aa972	2	12	9	2	9	9	17	1	06/04/20	29/03/20	9999-99-99	97	2	66	1	97
26	30/11/20	00f853	2	12	9	2	9	9	17	1	06/04/20	31/03/20	9999-99-99	97	2	29	1	97
27	30/11/20	1315ae	2	12	11	1	11	11	27	1	06/04/20	27/03/20	9999-99-99	97	2	56	1	2
28	30/11/20	06022d	1	6	25	2	25	25	6	2	11/04/20	28/03/20	9999-99-99	2	1	82	1	97
29	30/11/20	0025c6	2	12	28	2	15	28	38	1	04/04/20	30/03/20	9999-99-99	97	2	45	1	97
30	30/11/20	19c077	1	12	9	2	9	9	4	2	27/03/20	21/03/20	9999-99-99	2	2	57	1	97
31	30/11/20	1cc82a	2	6	27	2	27	27	2	1	27/03/20	25/03/20	9999-99-99	97	2	28	1	97

Figura 13. Conjunto de datos en formato CSV, obtenidos de la página del Gobierno de México.

4.2 Fase 2: Pre-procesamiento de la información

Se requiere obtener una representación estructurada y homogénea de la información recuperada, es por ello que se llevan a cabo algunas modificaciones necesarias en los datos y de esa manera, después se realizará el procesamiento de todos los registros de pacientes que han padecido COVID-19. En la sección 4.2.1 se describe el diseño e implementación de la base de datos en el DBMS MariaDB.

4.2.1 Implementación en MariaDB

Para llevar a cabo el pre-procesamiento de la información como primer paso es muy importante definir la base de datos que se implementará en MariaDB, es por ello que se realiza el diagrama entidad – relación correspondiente para poder crearla de acuerdo a todos los atributos que se estarán manejando (ver Figura 14).

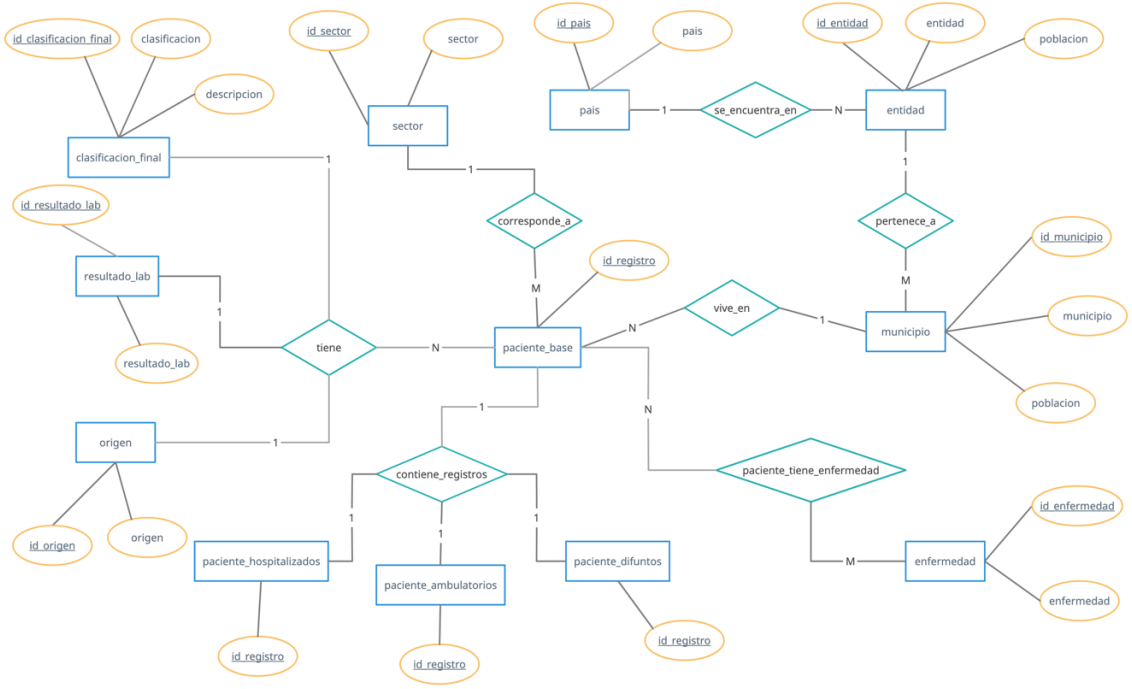


Figura 14. Diagrama entidad - relación para la base de datos COVID-19.

Como ya se ha mencionado anteriormente, esta base de datos se implementa en MariaDB, ya que es un DBMS (Sistema de Gestión de Bases de Datos) compatible con MySQL, además de que es notablemente escalable y brinda velocidad y alta seguridad en el almacenamiento de la información; de esta manera, mediante la estructura definida en el diagrama ER mostrado en la figura 16, se crea la BD llamada "covid19_pruebas" con cada una de las tablas definidas así como sus atributos con la estructura correspondiente a cada tipo de dato (ver Figura 15).

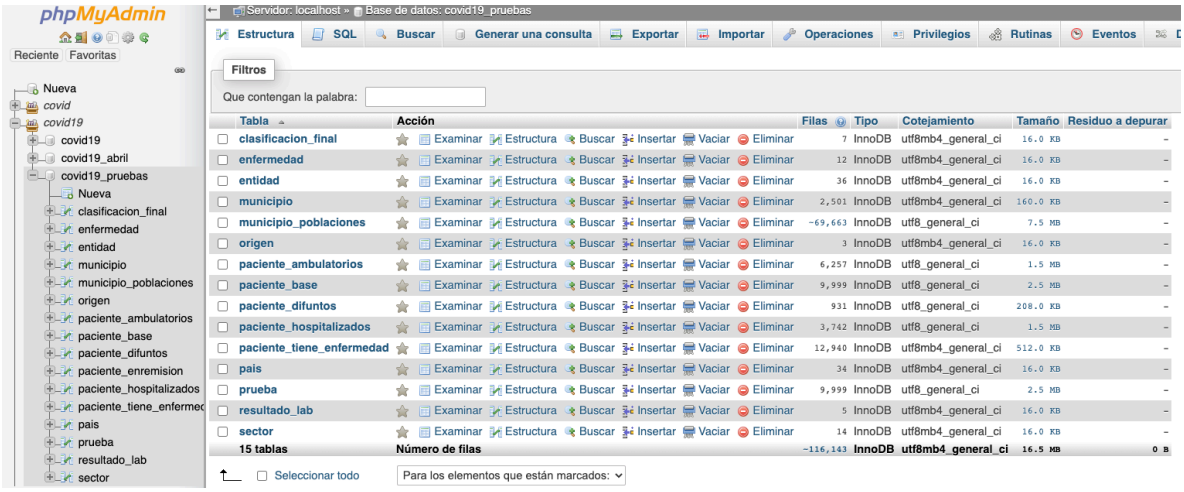


Figura 15. Base de datos "covid19_pruebas" implementada en MariaDB.

Una vez que se tiene creada toda la estructura de la BD para poder almacenar la información, esta se irán modificando mendiante Python, ya que este lenguaje de programación nos permite utilizar una librería conocida como "mariadb" para poder llevar a cabo la conexión correspondiente al servidor local de nuestra base de datos creada, de esta manera, mediante programación definiremos cada uno de los atributos que se almacenarán en dicha BD.

Nuestros archivos fuente serán los que se obtuvieron en la primera fase de búsqueda y recolección de información referente a los pacientes con COVID-19. Recordemos que estos archivos se encuentran en formato "CSV", por lo que Python también nos provee de herramientas propias del lenguaje las cuales leen este tipo de archivos para poder acceder a su contenido.

En la primera parte del script creado llamado "cargar_datos.py" se define la conexión con el servidor de MariaDB para poder acceder a la BD y hacer la inserción de los datos, después se lleva a cabo la lectura del archivo denominado "datos_covid_pruebas.csv" el cual contiene una muestra de 10 mil pacientes registrados con COVID-19.

Se define la tabla "paciente_base" como la correspondiente para insertar todos los datos con su nueva estructura y finalmente se definen todos los atributos por cada paciente registrado, de los cuales se insertarán en la tabla, es decir, el script se encargará de leer cada línea del archivo CSV (ver Figura 17), las cuales corresponden a cada paciente y cada atributo se asigna a la variable llamada "datos" la cual funge como un arreglo, cada uno de estos atributos serán insertados dependiendo la posición en que se definieron, esta estructura la definimos con Python para facilitar de cierta manera la inserción de grandes cantidades de información y no saturar el servidor de MariaDB, el script se presenta en la Figura 16.

```python
try:
    conn = mariadb.connect(
        user="root",
        password="",
        host="127.0.0.1",
        port=3306,
        database="covid19_actualizada"

    )
except mariadb.Error as e:
    print(f"Error connecting to MariaDB Platform: {e}")
    sys.exit(1)

# Get Cursor
cur = conn.cursor()
#csv=open("datos_covid_pruebas.csv",'r',encoding="ISO-8859-1") #para el archivo general
csv=open("datos_covid_pruebas.csv",'r',encoding="UTF-8")
#csv=open("datos2_covid_8mil.csv",'r',encoding="UTF-8")
flag=False
tabla="paciente_base"

for line in csv.readlines():
    if flag:
        try:
            line=line.replace("\n","")
            datos=line.split(",")

            cur.execute("INSERT INTO paciente_base (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACT
            ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)",
            (datos[1],datos[2],datos[3],datos[4],datos[5],datos[6],datos[7],datos[8],datos[9],datos[0],datos[10],datos[11],datos[12],datos[13],datos[15],datos[16],datos[17],datos[18]

            conn.commit()
        except mariadb.Error as e:
            print(f"Error: {e}")
    else:
        flag=True
        pass
print(f"Last Inserted ID: {cur.lastrowid}")
input()
conn.close()
```

Figura 16. Script realizado en Python para cargar los registros de pacientes con COVID-19 a la BD.

Settings | datos_covid_pruebas.csv

```
07/12/20,19c077,1,12,9,2,9,9,4,2,27/03/20,21/03/20,9999-99-99,2,2,57,1,97,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,2
07/12/20,1cc82a,2,6,27,2,27,27,2,1,27/03/20,25/03/20,9999-99-99,97,2,28,1,97,2,2,2,2,2,2,1,2,2,1,2,2,2,1,1,2,97,3,99,México,97,97
07/12/20,0ad0d2,1,12,9,2,15,15,37,1,01/04/20,27/03/20,9999-99-99,97,2,38,1,97,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,04431b,2,12,9,1,9,9,5,1,02/04/20,30/03/20,9999-99-99,97,2,22,1,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,97
07/12/20,19b1c8,2,12,31,1,31,31,50,1,20/03/20,10/03/20,9999-99-99,97,2,20,1,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,97
07/12/20,1c96f2,1,12,9,2,14,9,12,1,26/03/20,24/03/20,9999-99-99,97,2,37,1,97,2,2,2,2,2,2,2,1,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,1c7328,2,12,15,1,15,15,31,2,27/03/20,21/03/20,9999-99-99,2,1,33,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,2
07/12/20,07ddb4,2,9,21,1,21,21,119,1,28/03/20,24/03/20,9999-99-99,97,2,44,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,114f5f,2,4,9,2,12,9,6,1,31/03/20,29/03/20,9999-99-99,97,2,33,1,97,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,10a401,2,4,2,1,20,2,4,1,29/03/20,28/03/20,9999-99-99,97,2,32,1,2,2,1,2,2,2,2,2,2,2,2,2,1,99,1,1,2,97,3,99,México,97,97
07/12/20,184423,2,4,9,1,9,9,10,2,29/03/20,19/03/20,9999-99-99,2,1,48,1,2,2,2,2,2,1,2,1,2,2,2,2,2,2,99,1,1,2,97,3,99,México,97,2
07/12/20,0108cf,1,13,21,1,21,21,114,1,05/04/20,29/03/20,9999-99-99,97,2,53,1,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,14ee13,2,4,30,1,30,30,87,1,03/04/20,30/03/20,9999-99-99,97,2,51,1,2,2,2,2,2,2,1,2,2,2,2,2,2,99,1,1,2,97,3,99,México,97,97
07/12/20,0f0f91,1,6,9,2,9,9,17,2,07/04/20,25/03/20,19/05/20,2,1,62,1,97,2,2,1,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,1
07/12/20,06022d,1,6,25,2,25,25,6,2,11/04/20,28/03/20,9999-99-99,2,1,82,1,97,2,2,2,2,2,2,2,1,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,2
07/12/20,0025c6,2,12,28,2,15,28,38,1,04/04/20,30/03/20,9999-99-99,97,2,45,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,97
07/12/20,0fd0ba,1,12,9,1,15,15,104,2,02/04/20,21/03/20,17/04/20,1,1,65,1,2,2,2,2,2,2,1,1,1,2,2,2,2,1,1,1,2,97,3,99,México,97,1
07/12/20,13d662,2,12,9,1,9,9,10,1,04/04/20,20/03/20,9999-99-99,97,2,34,1,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,10811,1,9,9,1,99,9,10,1,18/03/20,13/03/20,9999-99-99,97,2,48,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,Macao,97,97
07/12/20,14cf92,1,13,21,2,30,21,114,1,29/03/20,29/03/20,9999-99-99,97,2,30,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,16d67b,1,12,25,1,25,25,6,2,03/04/20,01/03/20,03/04/20,2,1,52,1,2,2,2,2,2,2,2,1,2,2,2,2,2,2,1,1,2,97,3,99,México,97,2
07/12/20,1522c2,2,12,31,1,31,31,50,1,25/03/20,17/03/20,9999-99-99,97,2,59,1,2,2,2,2,1,2,2,1,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,17e0df,1,12,26,1,26,26,29,1,25/03/20,16/03/20,9999-99-99,97,2,21,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,2,1,1,2,97,3,99,México,97,97
07/12/20,063a56,2,12,31,2,31,31,50,1,15/03/20,09/03/20,9999-99-99,97,2,37,1,97,2,2,2,2,2,2,2,1,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,0ef6a2,1,12,9,1,9,15,24,1,28/03/20,27/03/20,9999-99-99,97,2,51,1,2,2,2,1,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,148214,2,4,2,2,2,2,4,2,31/03/20,26/03/20,9999-99-99,2,1,60,1,97,2,2,1,2,2,2,1,2,1,2,2,2,99,1,1,2,97,3,99,México,97,2
07/12/20,081e3e,1,4,2,1,2,2,2,2,30/03/20,29/03/20,19/04/20,2,1,62,1,2,2,2,2,2,1,2,2,1,2,2,2,1,1,99,1,1,2,97,3,99,México,97,2
07/12/20,111011,1,12,9,1,21,9,15,2,31/03/20,18/03/20,14/04/20,1,1,83,1,2,2,2,1,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,2
07/12/20,12d09d,2,9,9,2,9,9,7,2,05/04/20,30/03/20,9999-99-99,1,1,50,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,1
07/12/20,159ea5,2,9,8,2,8,8,37,1,13/04/20,30/03/20,9999-99-99,97,1,53,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,2,97,3,99,México,97,97
07/12/20,1.33E+08,1,4,3,2,3,3,8,1,02/04/20,27/03/20,9999-99-99,97,2,60,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,99,1,1,2,97,3,99,México,97,97
07/12/20,0fac67,2,12,1,2,11,1,1,1,22/03/20,18/03/20,9999-99-99,97,2,48,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,146404,2,12,9,2,28,28,38,1,22/03/20,21/03/20,20/04/20,97,1,77,1,97,2,2,1,2,2,2,1,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,0c9126,2,9,21,2,21,21,119,2,24/03/20,16/03/20,9999-99-99,2,2,25,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,México,97,2
07/12/20,02b16e,1,12,9,2,14,9,12,1,19/03/20,15/03/20,9999-99-99,97,2,25,1,97,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1,2,1,1,2,97,3,99,México,97,97
07/12/20,193721,2,12,1,2,99,1,6,1,27/03/20,26/03/20,9999-99-99,97,2,79,2,97,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1,2,97,3,99,Cuba,97,97
07/12/20,1267cc,2,12,21,1,21,21,19,1,26/03/20,22/03/20,9999-99-99,97,2,39,1,2,2,2,2,2,2,2,2,2,2,1,2,2,2,1,1,2,97,3,99,México,97,97
07/12/20,1dc009,1,13,21,2,20,21,114,1,29/03/20,26/03/20,9999-99-99,97,2,31,1,97,2,2,2,2,2,2,2,2,2,2,1,2,2,1,1,1,2,97,3,99,México,97,97
07/12/20,1de677,1,4,15,2,9,15,121,2,23/03/20,22/03/20,30/03/20,2,1,49,1,97,2,2,1,2,2,2,1,1,2,2,1,1,99,1,1,2,97,3,99,México,97,2
07/12/20,1b6dac,1,4,10,2,10,10,7,2,25/03/20,25/03/20,01/04/20,2,1,73,1,97,2,2,1,2,2,2,1,2,2,2,2,2,99,1,1,2,97,3,99,México,97,2
07/12/20,065bb1,2,4,9,1,15,15,58,1,31/03/20,31/03/20,9999-99-99,97,1,44,1,2,2,2,2,2,2,2,2,2,2,2,2,2,99,1,1,2,97,3,99,México,97,97
```

Figura 17. Archivo en formato CSV con la información que será procesada para insertar en la BD.

Siguiendo el proceso de estructuración de todos los datos, se realizaron algunas consultas específicas de la tabla “paciente_base”, se utilizó el lenguaje de consulta SQL para poder obtener los pacientes de acuerdo a su tipo (ambulatorios u hospitalizados) así como los pacientes difuntos, esta clasificación

del tipo de pacientes nos será de gran ayuda para después poder analizar la información del paciente y aplicar minería de datos para obtener un análisis preciso con algunas métricas que nos definan aspectos claves referentes a esta enfermedad (Figura 18).

```sql
-- Consultas para separar tipo de paciente ambulatorios
INSERT INTO `paciente_ambulatorios` (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION, FE
SELECT * FROM `paciente_base`
WHERE TIPO_PACIENTE = 1

--Pacientes Hospitalizado
INSERT INTO `paciente_hospitalizados` (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION,
SELECT * FROM `paciente_base`
WHERE TIPO_PACIENTE = 2

--Pacientes difuntos
INSERT INTO `paciente_difuntos` (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION, FECHA_
SELECT * FROM `paciente_base`
WHERE `FECHA_DEF` != "9999-99-99"

--Consultas para ubicacion
SELECT E.ENTIDAD, M.MUNICIPIO, S.SECTOR, PB.ID_REGISTRO FROM ((paciente_base as PB JOIN entidad as E ON ( PB.ID_ENTIDAD_RES = E.ID_ENTIDAD ) ) JOIN municipio as M on ( PB.ID_ENTIDAD
```

Figura 18. Consultas en SQL para la clasificación de acuerdo al tipo de paciente.

Para poder llevar al cabo el registro en la tabla "paciente_tiene_enfermedad", de la misma manera se crea un script de Python, en el cual se analiza cada registro (paciente) y el valor asignado en la enfermedad crónica que corresponda, 1 = Si y 2 = No, por lo tanto, en todos los registros con valor 1 se hará el INSERT a la tabla correspondiente como se muestra en la Figura 19.

```python
try:
    line=line.replace("\n","")
    datos=line.split(",")

    if (int(datos[14]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],1))
        pass
    if (int(datos[20]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],2))
        pass
    if (int(datos[21]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],3))
        pass
    if (int(datos[22]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],4))
        pass
    if (int(datos[23]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],5))
        pass
    if (int(datos[24]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],6))
        pass
    if (int(datos[25]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],7))
        pass
    if (int(datos[26]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],8))
        pass
    if (int(datos[27]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],9))
        pass
    if (int(datos[28]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],10))
        pass
    if (int(datos[29]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],11))
        pass
    if (int(datos[30]) == 1):
        cur.execute("INSERT INTO paciente_tiene_enfermedad (ID_REGISTRO, ID_ENFERMEDAD) VALUES (?, ?)", (datos[1],12))
        pass
```

Figura 19. Script para la inserción de las enfermedades crónicas y sintomatologías que presenta cada paciente además del COVID-19.

Una vez organizada la información por el tipo de paciente y cada registro con todos sus atributos, se realiza un cálculo para obtener la ubicación de cada paciente, tomando en cuenta el sector al que pertenecen para conocer el lugar donde recibieron atención médica; todo esto se lleva a cabo para obtener otros valores los cuales nos permitan analizarlos y obtener información nueva que nos ayude a identificar patrones clave que se están originando en el contexto de la pandemia a causa de COVID-19.

En el script programado en Python mostrado en la Figura 20, se presenta como se lleva a cabo el proceso para calcular la ubicación de cada uno de los pacientes; se tienen que evaluar la entidad, el municio y el sector al que pertenecen cada uno, para que así se pueda realizar una consulta en SQL para extraer los valores correspondientes y después formar una cadena la cual nos será de gran ayuda para poder enviar esos parámetros a Google Maps mediante el uso de la API correspondiente y como resultado nos devolverá la ubicación correcta que corresponda a dichos datos.

```
cur.execute("SELECT ID_ENTIDAD_RES, ID_MUNICIPIO_RES, ID_SECTOR, ID_REGISTRO FROM paciente_base")
for (entidad, municipio, sector, id) in (cur):
    flag=False
    if (int(entidad) != 97 and int(entidad) != 98 and int(entidad) != 99):
        if (int(municipio) != 997 and int(municipio) != 998 and int(municipio) != 999 ):
            if (int(sector) != 97 and int(sector) != 98 and int(sector) != 99 ):
                print("-------",i)
                i+=1
                try:
                    cur3.execute("SELECT ENTIDAD FROM entidad WHERE CAST('"+str(entidad)+"' AS int) =  CAST(ID_ENTIDAD AS int)")
                    entidad_temp = (cur3.fetchone()[0])

                    cur3.execute("SELECT MUNICIPIO FROM municipio WHERE CAST('"+str(municipio)+"' AS int) =  CAST(ID_MUNICIPIO AS int) AND CAST('"+str(entidad)+"' AS int) = CAST(ID_
                    municipio_temp = (cur3.fetchone()[0])

                    cur3.execute("SELECT SECTOR FROM sector WHERE CAST('"+str(sector)+"' AS int) =  CAST(ID_SECTOR AS int)")
                    sector_temp = (cur3.fetchone()[0])
                    pass
                except TypeError as e:
                    flag=True
                if flag:
                    print ("Error en el catalogo")
                    #Eliminar La columna en el dado caso DELETE
                else:
                    entidad=normalize(entidad_temp)
                    municipio=normalize(municipio_temp)
                    sector=normalize(sector_temp)
                    entidad=entidad.replace(" ","%20")
                    municipio=municipio.replace(" ","%20")
                    sector=sector.replace(" ","%20")
                    union=str(sector)+str(entidad)+str(municipio)
                    if union not in geo:
                        googleGeocodeUrl = 'https://maps.googleapis.com/maps/api/place/textsearch/json?query='
                        termino = str(entidad)+"%20"+str(municipio)+"%20hospital%20"+str(sector)
                        APIKEY = '&key='+'AIzaSyDp5krNVNh8I6Uy4Q6Vx_-iW4AgdMx9944'
                        url = googleGeocodeUrl + termino + APIKEY
                        urllib.parse.quote(url)
                        time.sleep(1)
                        geo.append(union)
                        json_response = urllib.request.urlopen(url)
```

Figura 20. Programa en Python que se encarga de realizar la obtención de cada uno de los pacientes registrados.

Para este caso hacemos uso del tipo de dato espacial “POINT”, el cual nos permite definir dos valores (X y Y), en este caso los manejaríamos como la latitud y la longitud que son las coordenadas que nos devuelve Google Maps al obtener la ubicación. Una vez que se obtuvo este nuevo atributo, mediante Python se lleva a cabo un update a la base de datos para poder insertar los nuevos registros en la columna correspondiente llamada “Ubicación” que se encuentra en la tabla principal conocida como “paciente_base”, las funciones que realizan estas actualizaciones en los datos se muestran en la Figura 21.

```
    busqueda = json_response.read().decode('utf-8')
    busquedajson = json.loads(busqueda)
    try:
        lat=busquedajson['results'][0]['geometry']['location']['lat']
        lng=busquedajson['results'][0]['geometry']['location']['lng']
        print (lat, lng)
        ubicacion=str(lat)+","+str(lng)
        data.append(ubicacion)
        print("Guardando Datos:")
        guardar(geo,data)
        print("Listo")
        pass
    except (KeyError,IndexError) as e:
        cur2 = conn2.cursor()
        sql="Update paciente_base set UBICACION=POINT(0,0) WHERE ID_REGISTRO='"+id+"';"
        print(sql)
        cur2.execute(sql)
        conn2.commit()
        data.append("0,0")
        print ("Algo paso")
        pass
    #Empieza Update
    cur2 = conn2.cursor()
    sql="Update paciente_base set UBICACION=POINT("+ubicacion+") WHERE ID_REGISTRO='"+id+"';"
    print(sql)
    cur2.execute(sql)
    conn2.commit()
    # input("terminé -- usó API")
else:
    ubicacion=data[geo.index(union)]
    print ("No Use API", ubicacion)
    #Empieza Update
    cur2 = conn2.cursor()
    sql="Update paciente_base set UBICACION=POINT("+ubicacion+") WHERE ID_REGISTRO='"+id+"';"
    print(sql)
    cur2.execute(sql)
    conn2.commit()
```

Figura 21. Código en Python que se encarga de realizar el update a la BD para insertar la ubicación de cada registro.

4.3 Fase 3: Procesamiento de datos

Para llevar a cabo el procesamiento de la información, se realiza la implementación en un motor de búsqueda conocido como ElasticSearch, el cual permite hacer búsquedas casi en tiempo real, es de alto rendimiento, permite el almacenamiento de documentos sin formato, este almacenamiento lo realiza en la nube por lo que permite insertar grandes cantidades de información. ElasticSearch trabaja en conjunto con Kibana, la cual es la interfaz para permitir la interacción con este motor de búsqueda, además de utilizar el lenguaje llamado Logstash para realizar scripts y poder ejecutarlos para llevar a cabo las inserciones de los datos dentro del núcleo de este software.

En la sección 4.3.1 se describen los pasos para desarrollar la BD en ElasticSearch; en la sección 4.3.2 se presentan las modificaciones requeridas a la BD en MariaDB para agregar nuevos atributos en la información, mientras que en la sección 4.3.3 se lleva a cabo la actualización de la base de datos en este mismo servidor para tener los últimos datos registrados. Y finalmente, en la sección 4.3.4 se actualiza la información en el motor de búsqueda ElasticSearch.

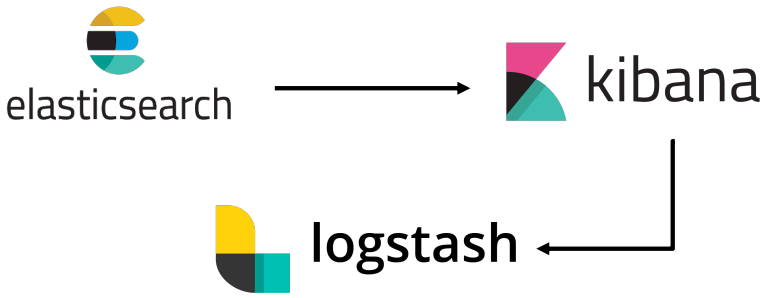


Figura 22. Herramientas de software con las que trabaja en conjunto ElasticSearch.

4.3.1 Implementación en ElasticSearch

Iniciando con el procesamiento de los datos, se ha realizado un script en lenguaje Logstash, el cual se encarga de realizar la conexión con el servidor de ElasticSearch, similar a Python, este programa se encarga de leer un archivo fuente en formato CSV, el cual se obtiene de la base de datos creada en el pre-procesamiento de la información, ya que en esa fase fue donde se le dio la estructura homogénea a los datos; así en esta tercera fase podemos hacer la inserción a ElasticSearch ya con el conjunto de datos modificado a nuestras necesidades.

Se deben definir cada uno de los atributos que se insertarán en el servidor para que los registros puedan ser leídos en el archivo fuente, además de que es muy importante dar el formato adecuado a los atributos de fechas, ya que este motor de búsqueda nos permite seleccionar un atributo como pivote y así hacer una clasificación de los datos, para esta implementación se eligió la fecha en que presentaron síntomas los pacientes y así poder observar como se están agrupando por mes todos los registros insertados. En la Figura 23 se muestra parte del script creado para llevar a cabo la inserción de datos en ElasticSearch.

```
input {

  file {
    path => "/Users/luisyaelms/Documents/Elk/logstash-7.10.0/covidfcc/datos_pb_covid.csv"
    start_position => "beginning"
  }
}

filter{

  mutate {
    gsub => [ "message", "\r", "" ]
  }

  dissect {
    mapping => { "message" => "%{id_registro},%{origen},%{sector},%{entidad_um},%{sexo},%{entidad_nac},%{entidad_res},%{municipio_res},%{tipo_paciente},%{fecha_actualizacion},%{fecha
  }

  mutate {
      gsub => ["fecha_def", "9999-99-99","01/01/99"]
  }

  date {
      match => [ "fecha_ingreso", "dd/MM/yy" ]
      target => "f_ingreso"
  }

  date {
      match => [ "fecha_def", "dd/MM/yy" ]
      target => "f_def"
  }

  date {
      match => [ "fecha_sintomas", "dd/MM/yy" ]
      target => "@timestamp"
  }
```

Figura 23. Script en lenguaje Logstash para insertar datos en el servidor de ElasticSearch.

Después de especificar todos los atributos es necesario definir también su tipo de dato para que al insertarlos no haya problemas, uno por uno se deben ir asignando y así el software interpretará y creará el archivo correspondiente para la inserción; por otra parte también se define la salida al servidor de ElasticSearch para enviar todos los datos que se van a almacenar. Además se tiene que especificar el nombre del índice con el que aparecerá en la interfaz de Kibana, es decir, el index es el nombre que se le da a la base de datos, solo que en esta herramienta de software se manejan como índices ya que son datos no estructurados (ver Figura 24).

```
mutate {
  convert   => {
                "id_registro" => "string"
                "origen" => "integer"
                "sector" => "integer"
                "entidad_um" => "integer"
                "sexo" => "integer"
                "entidad_nac" => "integer"
                "entidad_res" => "integer"
                "municipio_res" => "integer"
                "tipo_paciente" => "integer"
                "intubado" => "integer"
                "edad" => "integer"
                "nacionalidad" => "integer"
                "embarazo" => "integer"
                "habla_lengua_indig" => "integer"
                "indigena" => "integer"
                "toma_muestra" => "integer"
                "resultado_lab" => "integer"
                "toma_muestra_antigeno" => "integer"
                "resultado_antigeno" => "integer"
                "clasificacion_final" => "integer"
                "migrante" => "integer"
                "pais_nacionalidad" => "string"
                "pais_origen" => "string"
                "uci" => "integer"
                "latitud" => "string"
                "longitud" => "string"

              }
    remove_field => [ "message", "fecha_def", "fecha_sintomas"]
  }
}
output {

  elasticsearch {
    hosts => ["localhost:9200"]
    index => "datos-covid-ubicacion-%{+YYYY.MM}"
  }
  stdout { codec => rubydebug }
```

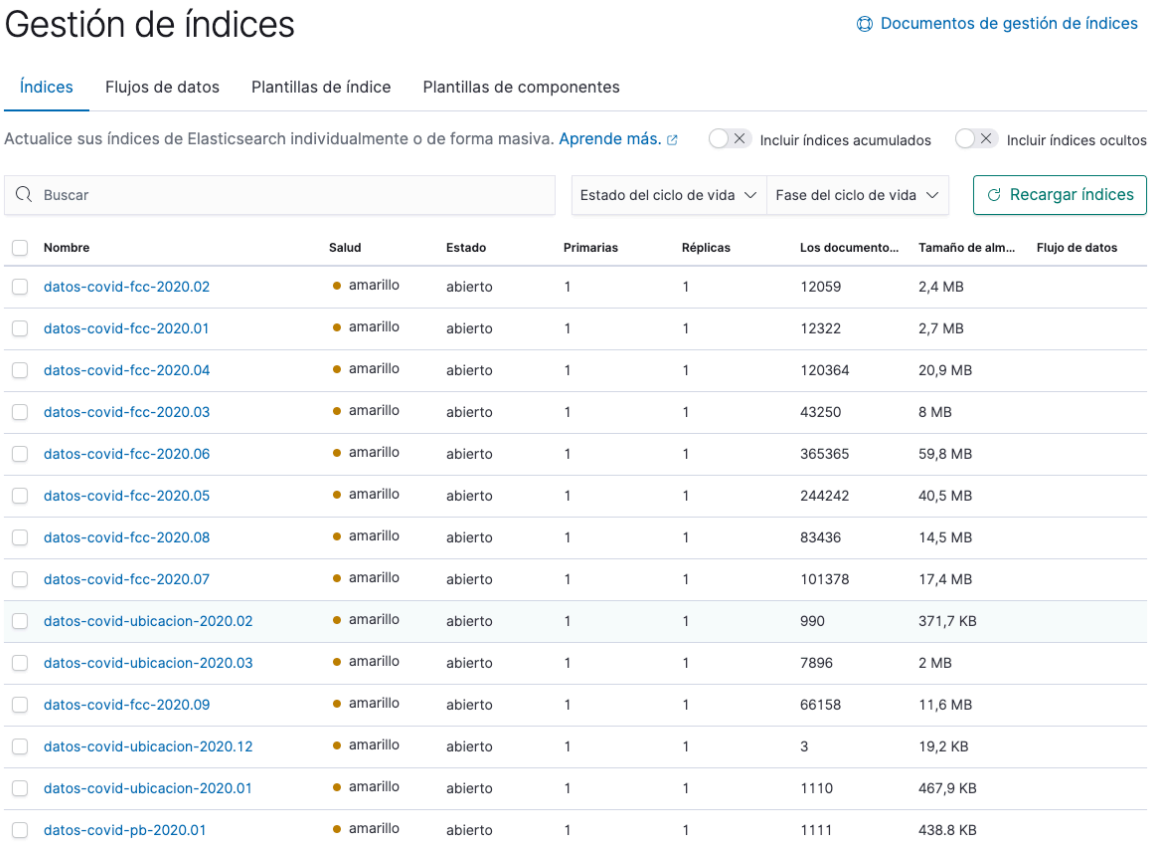
Figura 24. Definición de tipos de dato por cada atributo y salida al servidor de ElasticSearch en lenguaje Logstash.

Una vez que se ejecuta el script de Logstash para insertar los datos, se crea una estructura de datos conocida como "JSON", en la cual se ordenan todos los atributos de acuerdo a cada paciente registrado, en este caso por cada paciente se creará un json que servirá para llevar a cabo el almacenamiento de la información, ya que se transfiere mediante la conexión al servidor de ElasticSearch, en la Figura 25 se muestra el ejemplo de un json creado con los datos de un registro completo.

```
                "host" => "MacBook-Pro-de-Luis-2.local",
          "id_registro" => "2",
            "embarazo" => 2,
          "f_ingreso" => 2020-07-24T05:00:00.000Z,
       "resultado_lab" => 1,
               "epoc" => 2,
         "@timestamp" => 2020-07-18T05:00:00.000Z,
"toma_muestra_antigeno" => 2,
                "uci" => 97,
      "dias_atenderse" => 6,
       "renal_cronica" => 2,
          "tabaquismo" => 2,
  "pais_nacionalidad" => "México",
        "entidad_nac" => 4,
           "intubado" => 97,
 "fecha_actualizacion" => "45503",
              "sexo" => 4,
              "edad" => 1,
            "sector" => 4,
    "estado_paciente" => "R",
         "otro_caso" => 1,
       "entidad_res" => 6,
            "origen" => 12,
          "diabetes" => 2,
      "tipo_paciente" => 31,
 "resultado_antigeno" => 97,
      "fecha_ingreso" => "24/07/20",
       "nacionalidad" => 97,
           "migrante" => 99,
               "path" => "/Users/luisyaelms/Documents/Elk/logstash-7.10.0/covidfcc/importacion.csv",
         "entidad_um" => 2,
          "indigena" => 2,
    "cardiovascular" => 2,
         "inmusupr" => 2,
          "obesidad" => 1,
      "hipertension" => 2,
     "municipio_res" => 1,
          "otra_com" => 2,
            "f_def" => 1999-01-01T06:00:00.000Z,
 "habla_lengua_indig" => 2,
             "asma" => 2,
      "toma_muestra" => 1,
 "clasificacion_final" => 3,
          "@version" => "1",
       "pais_origen" => 97,
          "neumonia" => 35
```

Figura 25. JSON creado por ElasticSearch para hacer la inserción de los datos dentro de su servidor.

Una vez que se haya completado el almacenamiento de los datos en el servidor de ElasticSearch, se debe ejecutar la interfaz de Kibana, es importante llevar a cabo la creación de los índices de acuerdo al mes al que pertenece cada registro, en la congifuración de Kibana se accede a la gestión de índices y deberán aparecer todos los grupos creados de acuerdo a la fecha de síntomas como se muestra en la Figura 26.



Nombre	Salud	Estado	Primarias	Réplicas	Los documento...	Tamaño de alm...	Flujo de datos
datos-covid-fcc-2020.02	amarillo	abierto	1	1	12059	2,4 MB	
datos-covid-fcc-2020.01	amarillo	abierto	1	1	12322	2,7 MB	
datos-covid-fcc-2020.04	amarillo	abierto	1	1	120364	20,9 MB	
datos-covid-fcc-2020.03	amarillo	abierto	1	1	43250	8 MB	
datos-covid-fcc-2020.06	amarillo	abierto	1	1	365365	59,8 MB	
datos-covid-fcc-2020.05	amarillo	abierto	1	1	244242	40,5 MB	
datos-covid-fcc-2020.08	amarillo	abierto	1	1	83436	14,5 MB	
datos-covid-fcc-2020.07	amarillo	abierto	1	1	101378	17,4 MB	
datos-covid-ubicacion-2020.02	amarillo	abierto	1	1	990	371,7 KB	
datos-covid-ubicacion-2020.03	amarillo	abierto	1	1	7896	2 MB	
datos-covid-fcc-2020.09	amarillo	abierto	1	1	66158	11,6 MB	
datos-covid-ubicacion-2020.12	amarillo	abierto	1	1	3	19,2 KB	
datos-covid-ubicacion-2020.01	amarillo	abierto	1	1	1110	467,9 KB	
datos-covid-pb-2020.01	amarillo	abierto	1	1	1111	438.8 KB	

Figura 26. Gestión de índices en la interfaz de Kibana.

Por cada grupo de índices de acuerdo al mes se crea un patrón de índice que será el que permita acceder a la información y poder trabajar con ella desde la interfaz de Kibana, es decir, una vez creado este patrón de índice se podrán acceder a todos los datos para interactuar con ellos (Figura 27).

Figura 27. Patrón de índice "datos-covid-pb" creado para trabajar con todos los datos insertados en ElasticSearch.

Una vez que se ha realizado todo ese proceso de gestión desde Kibana, se podrá acceder a la base de datos creada en este motor de búsqueda y se clasificará por grupos de acuerdo al mes en que ha sufrido síntomas de COVID-19 como se muestra en la Figura 28.

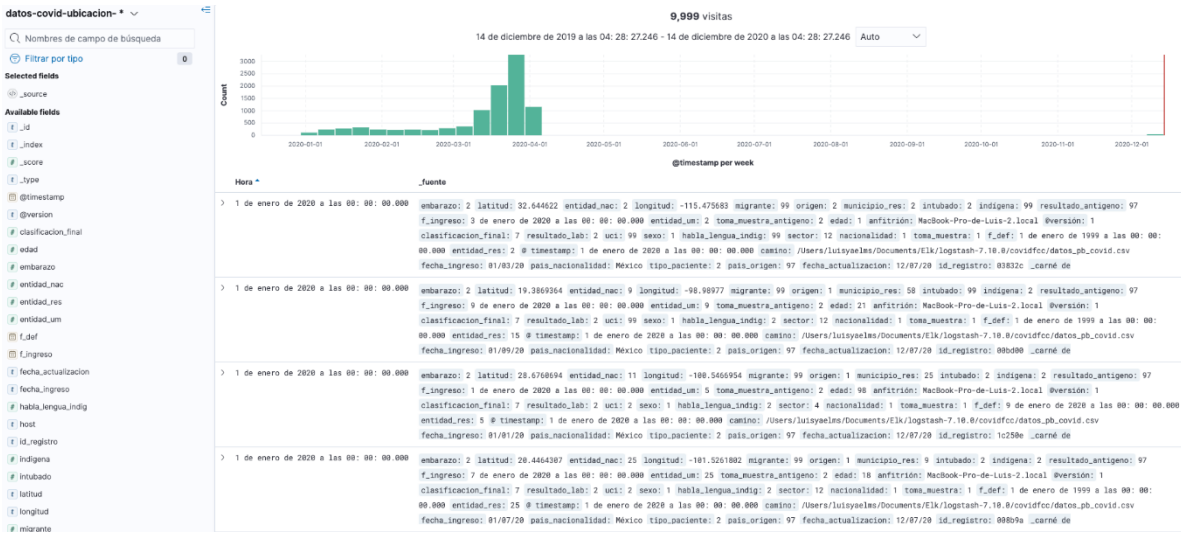


Figura 28. Base de datos de COVID-19 creada en ElasticSearch.

4.3.2 Modificaciones en MariaDB

En la fase de procesamiento de los datos, también se llevaron a cabo alguna modificaciones importantes en la base de datos de COVID-19; se hicieron algunas consultas en SQL para poder separar la información de acuerdo a los pacientes que se encuentran en remisión, así como los pacientes en defunción, esto para poder clasificarlos y hacer un análisis de los diferentes factores que puedieron haber influído en esos resultados. Las consultas realizadas se muestran en la Figura 29 y Figura 30.

```sql
--Pacientes difuntos
INSERT INTO `paciente_difuntos` (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION,
  FECHA_INGRESO, FECHA_SINTOMAS, FECHA_DEF, INTUBADO, EDAD, NACIONALIDAD, EMBARAZO, HABLA_LENGUA_INDIG, INDIGENA, TOMA_MUESTRA, ID_RESULTADO_LAB, TOMA_MUESTRA_ANTIGENO,
  ID_RESULTADO_ANTIGENO, ID_CLASIFICACION_FINAL, MIGRANTE, ID_PAIS_NACIONALIDAD, ID_PAIS_ORIGEN, UCI)
SELECT * FROM `paciente_base`
WHERE `FECHA_DEF` != "9999-99-99"
```

Figura 29. Consulta en SQL para clasificar a los pacientes por defunción.

```sql
--Insertar pacientes en remision
INSERT INTO paciente_enremision SELECT * FROM paciente_sin_ubicacion
WHERE FECHA_DEF = "9999-99-99";
```

Figura 30. Consulta en SQL para clasificar los pacientes en remisión.

De igual forma, ha sido necesario crear una tabla auxiliar similar a paciente_base pero esta vez excluyendo el atributo de ubicación para después poder cargarlo en el software conocido como "Weka" y así realizar la aplicación de minería de datos con toda la información que se está trabajando. Se insertan todos los campos de la tabla original y finalmente se elimina el atributo ubicación de la tabla auxiliar que se creó, en la Figura 31 se muestra la pequeña consulta para realizar este proceso.

```sql
--Duplicar tabla paciente_base
CREATE TABLE paciente_sin_ubicacion LIKE paciente_base;

--Insertar datos de una tabla a otra
INSERT INTO paciente_sin_ubicacion SELECT * FROM paciente_base;
```

Figura 31. Consulta en SQL para crear la tabla auxiliar "paciente_sin_ubicacion" en la BD.

Una vez realizados estos cambios significativos en la base de datos para obtener una mejor clasificación y poder hacer un análisis con los datos mejor organizados, obtenemos como resultado la BD de COVID-19 con las siguientes tablas como se muestra en la Figura 32.

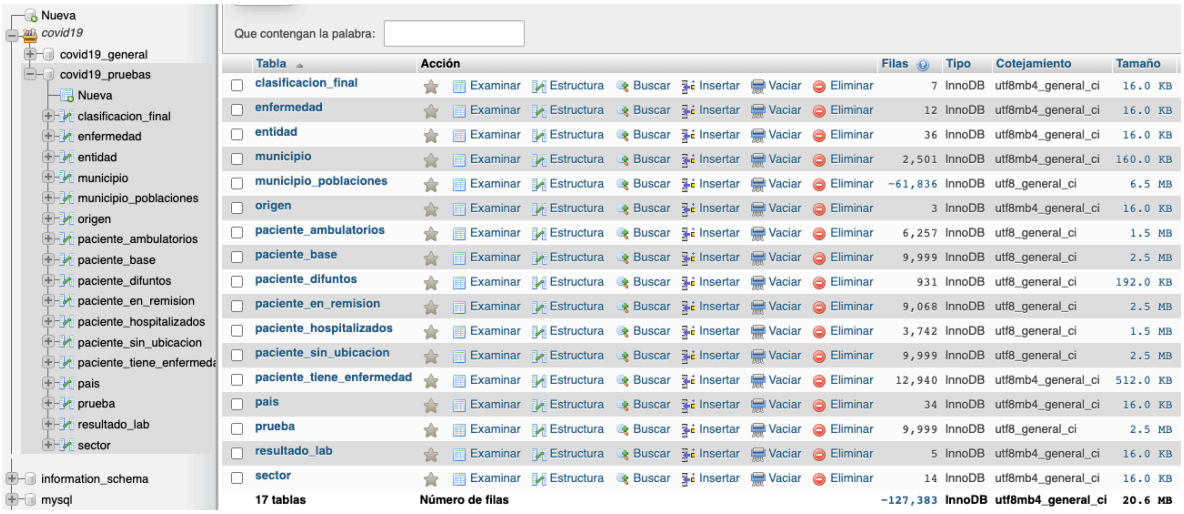


Figura 32. Base de datos "covid19_pruebas" con nuevas tablas creadas.

Ya que se tienen las nuevas tablas para mejorar la clasificación de la información, se realizaron algunos cálculos de acuerdo al transcurso de días que han tardado los pacientes en atenderse o bien en el caso de los pacientes difuntos, cuantos días fue que tardaron con esta lamentable enfermedad; estas operaciones se hacen mediante programación en Python por lo que, se realizaron dos scrips claves para llevar a cabo estas operaciones en cuanto a las fechas de síntomas, de ingreso y defunción. Ver Figura 33 y Figura 34.

```python
# Get Cursor
cur = conn.cursor()
cur.execute("SELECT ID_REGISTRO, FECHA_INGRESO, FECHA_SINTOMAS FROM pacientes_clasificados")
for (id_registro, fechaIngreso, fechaSintomas) in tqdm(cur):
    # Se transforma la fecha de ingreso
    fechaInicio=arreglarfechas(fechaIngreso)
    fecha_ingreso = datetime.strptime(fechaInicio, '%d/%m/%Y')
    #Se transforma la fecha de defunsion
    fechaFin=arreglarfechas(fechaSintomas)
    fecha_sintomas = datetime.strptime(fechaFin, '%d/%m/%Y')

    dias_t=(fecha_ingreso - fecha_sintomas)
    dias_transcurridos=int(dias_t.days)
    #print("Dias transcurridos: "+ dias_transcurridos)
    #print("Fecha ingreso: "+str(fecha_dt)+"Fecha Def: "+str(fecha_dt2))
    cur2 = conn2.cursor()
    sql="Update pacientes_clasificados set DIAS_ATENDERSE='"+str(dias_transcurridos)+"' WHERE ID_REGISTRO='"+id_registro+"';"

    # print(sql)
    cur2.execute(sql)
    conn2.commit()
    pass
conn.commit()
```

Figura 33. Script en Python para calcular el número de días que tardaron en atenderse los pacientes en remisión.

```
# Get Cursor
cur = conn.cursor()
#csv=open("datos_covid_pruebas.csv",'r',encoding="ISO-8859-1") #para el archivo general
csv=open("covid19_full_data.csv",'r',encoding="UTF-8")
#csv=open("datos2_covid_8mil.csv",'r',encoding="UTF-8")
flag=False
tabla="paciente_difuntos"

cur.execute("SELECT ID_REGISTRO, FECHA_INGRESO, FECHA_DEF FROM paciente_difuntos")
for (id_registro, fechaIngreso, fechaDef) in (cur):
    # Se transforma la fecha de ingreso
    fechaInicio=arreglarfechas(fechaIngreso)
    fecha_inicio = datetime.strptime(fechaInicio, '%d/%m/%Y')
    #Se transforma la fecha de defunsion
    fechaFin=arreglarfechas(fechaDef)
    fecha_fin = datetime.strptime(fechaFin, '%d/%m/%Y')

    dias_t=(fecha_fin - fecha_inicio)
    dias_transcurridos=int(dias_t.days)

    cur2 = conn2.cursor()
    sql="Update paciente_difuntos set DIAS_HOSPITALIZADOS='"+str(dias_transcurridos)+"' WHERE ID_REGISTRO='"+id_registro+"';"
    print(sql)
    cur2.execute(sql)
    conn2.commit()
    pass
conn.commit()
```

Figura 34. Script que calcula el número de días de hospitalización que tardaron los pacientes antes de su defunción.

Como ejemplificación, en la Figura 35 se muestran los días que transcurrieron en atenderse los pacientes en remisión, agregando una columna adicional al final de la tabla; mientras que en la Figura 36 se muestra como se insertó de la misma manera que el caso mencionado anteriormente la columna de días hospitalizados para los pacientes que lamentablemente fallecieron a causa del COVID-19.

ID_CLASIFICACION_FINAL	MIGRANTE	ID_PAIS_NACIONALIDAD	ID_PAIS_ORIGEN	UCI	DIAS_ATENDERSE
3	99	México	97	97	2
3	99	México	97	97	5
3	99	México	97	97	3
3	99	México	97	97	10
3	99	México	97	97	2
3	99	México	97	2	6
3	99	México	97	97	4
3	99	México	97	97	2
3	99	México	97	97	1
3	99	México	97	2	10
3	99	México	97	97	7
3	99	México	97	97	4
3	99	México	97	2	14
3	99	México	97	97	5
3	99	México	97	97	15
3	99	Macao	97	97	5
3	99	México	97	97	0
3	99	México	97	97	8
3	99	México	97	97	9
3	99	México	97	97	6
3	99	México	97	97	1
3	99	México	97	2	5
3	99	México	97	1	6
3	99	México	97	97	14
3	99	México	97	97	6

Figura 35. Muestra de la tabla pacientes en remisión para distinguir la columna "dias_atenderse".

ID_CLASIFICACION_FINAL	MIGRANTE	ID_PAIS_NACIONALIDAD	ID_PAIS_ORIGEN	UCI	DIAS_HOSPITALIZADOS
3	99	México	97	1	42
3	99	México	97	1	15
3	99	México	97	2	0
3	99	México	97	2	20
3	99	México	97	2	14
3	99	México	97	97	29
3	99	México	97	2	7
3	99	México	97	2	7
3	99	México	97	1	7
3	99	México	97	2	56
3	99	México	97	1	4
3	99	México	97	2	5
3	99	México	97	97	7
3	99	México	97	2	4
3	99	México	97	2	8
3	99	México	97	2	7
3	99	México	97	2	6
3	99	México	97	2	4
3	99	México	97	2	29
3	99	México	97	2	22
3	99	México	97	2	8
3	99	México	97	2	2
3	99	México	97	1	3
3	99	México	97	2	7
3	99	México	97	1	2

Figura 36. Muestra de la tabla de pacientes difuntos, para distinguir la columna "dias_hospitalizados".

4.3.3 Actualización de la base de datos en MariaDB

Antes de proceder a la última fase, es necesario llevar a cabo la actualización de la base de datos en el servidor local de MariaDB, para ello, utilizando el repositorio de datos de la Secretaría de Salud del Gobierno de México, obtenemos el último dataset actualizado que corresponde a los datos registrados hasta el 31 de mayo de 2021, el cual consta de 1 048 575 registros de pacientes. Además, para esta nueva base de datos será necesario agregar en la tabla “paciente_base” las columnas correspondientes a cada una de las comorbilidades que presentan los pacientes registrados, ya que nos serán útiles para analizar algunas características en los datos.

Inicialmente, se llevó a cabo la modificación del script en Python que se encarga de insertar todos los datos, los cuales se cargan desde el archivo fuente en formato CSV llamado “covid19_con_enfermedades”; después se tuvo que modificar la consulta en SQL que se encargar de realizar el INSERT a la base de datos, se agregaron los diferentes atributos para especificar cada una de las comorbilidades.

Una vez que se tienen especificados todos los atributos, se procede a ejecutar el script para que realice la inserción. La función principal para cargar los datos se muestra en la Figura 37.

```
# Get Cursor
cur = conn.cursor()
#csv=open("datos_covid_pruebas.csv",'r',encoding="ISO-8859-1") #para el archivo general
csv=open("covid19_con_enfermedades.csv",'r',encoding="UTF-8")
#csv=open("datos2_covid_8mil.csv",'r',encoding="UTF-8")
flag=True
tabla="paciente_base"

for line in tqdm(csv.readlines(),ascii=True, desc="Importando:"):
    if flag:
        try:
            line=line.replace("\n","")
            datos=line.split(",")
            sql=f"INSERT INTO paciente_base (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALI
            RESULTADO_ANTIGENO, CLASIFICACION_FINAL, MIGRANTE, ID_PAIS_NACIONALIDAD, ID_PAIS_ORIGEN, UCI) VALUES ('{datos[1]}', {datos[2]}, {datos[3]}, '{datos[4]}', {datos[5]}, '
            {datos[29]}, {datos[30]}, {datos[31]}, {datos[32]}, {datos[33]}, {datos[34]}, {datos[35]}, {datos[36]}, '{datos[37]}', '{datos[38]}', {datos[39]})"
            # print (sql)
            # input()
            cur.execute(sql)
            # cur.execute("INSERT INTO pacientes_clasificados (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIE
            ESTADO_PACIENTE) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,?)",
            # (str(datos[0]),datos[1],datos[2],str(datos[3]),datos[4],str(datos[5]),str(datos[6]),str(datos[7]),datos[8],str(datos[9]),str(datos[10]),str(datos[11]),str(datos[12])

            conn.commit()
        except mariadb.Error as e:
            print(f"Error: {e}")
    else:
        flag=True
        pass
print(f"Last Inserted ID: {cur.lastrowid}")
input()
conn.close()
```

Figura 37. Script en Python actualizado para cargar los registros de pacientes con comorbilidades.

Se crean de nuevo las tablas para clasificar los pacientes por ambulatorios u hospitalizados, así como los que se encuentran en remisión y los pacientes difuntos, ya que estas nuevas tablas deberán tener la misma estructura con la que se actualizó la tabla paciente_base. Después de crear las tablas, mediante SQL se implementan las consultas correspondientes para insertar los registros de pacientes de acuerdo a la clasificación especificada (ver Figura 38).

```sql
--Insertar pacientes ambulatorios en la BD covid19_act_comorbilidades
INSERT INTO pacientes_ambulatorios SELECT * FROM paciente_base
WHERE TIPO_PACIENTE = 1;

--Insertar pacientes hospitalizados
INSERT INTO pacientes_hospitalizados SELECT * FROM paciente_base
WHERE TIPO_PACIENTE = 2;

--Insertar pacientes en remisión
INSERT INTO pacientes_en_remision SELECT * FROM paciente_base
WHERE FECHA_DEF = "9999-99-99";

--Insertar pacientes difuntos
INSERT INTO pacientes_difuntos SELECT * FROM paciente_base
WHERE FECHA_DEF != "9999-99-99";
```

Figura 38. Consultas en SQL para insertar los datos de acuerdo al criterio de clasificación.

Una vez que se han actualizado cada una de las tablas con los nuevos registros de pacientes, esta base de datos será la que se utilice para llevar a cabo la conexión con el software conocido como WEKA y así iniciar con los análisis aplicando mineria de datos. En la Figura 39, se muestra la base de datos que se creó finalmente con las tablas y la información actualizada de pacientes registrados.

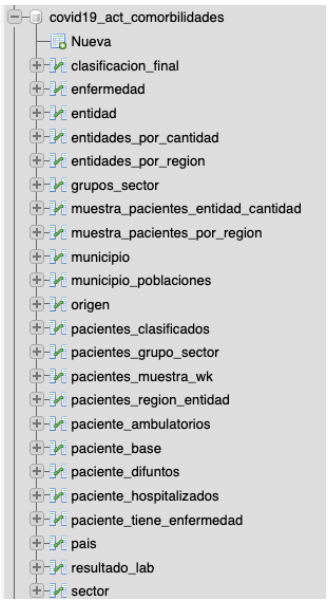


Figura 39. BD "covid19_act_comorbilidades" con las tablas e información actualizada al 31 de mayo de 2021.

4.3.4 Actualización de la base de datos en ElasticSearch

De igual forma, una vez que se actualizó la base de datos en el servidor local de MariaDB, sabemos que los datos se encuentran con la estructura homogénea que se requiere, es por ello que ahora, se lleva a cabo la actualización en el servidor de ElasticSearch, ya que esta base de datos será la más actual que se tiene, como ya se mencionó anteriormente, los registros de pacientes corresponden hasta la fecha del 31 de mayo de 2021, siendo un total de 1 048 574 pacientes registrados con COVID 19 por la Secretaría de Salud en México.

Es importante tener la información en este motor de búsqueda, ya que precisamente una de sus ventajas es que nos permite trabajar con grandes volúmenes de información, además de que seguimos trabajando con el atributo de la fecha de síntomas como pivote para poder ver la agrupación por mes y cuántos pacientes han presentado síntomas en determinadas fechas.

Para llevar a cabo la inserción de los datos, es necesario modificar el script que tenemos en lenguaje Logstash; cambiamos el nombre del archivo fuente que será leído desde línea de comandos como el PATH, el cual es la ruta especificada de donde se encuentra el dataset (ver Figura 40).

```
input {

  file {
    path => "/Users/luisyaelms/Documents/Elk/logstash-7.10.0/covidfcc/covid19_con_enfermedades.csv"
    start_position => "beginning"
  }
}
```

Figura 40. Se indica la ruta del PATH donde se encuentra el archivo con los registros de pacientes.

Después, en la sección conocida como “filter” se lleva a cabo la definición de cada uno de los encabezados que se van a insertar para indentificar los datos, es decir, aquí nombramos cada uno de los atributos específicos de cada paciente (ver Figura 41).

```
filter{

  mutate {
    gsub => [ "message", "\r", "" ]
  }

  dissect {
  #fecha_actualizacion,id_registro,origen,sector,entidad_um,sexo,entidad_nac,entidad_res,municipio_res,tipo_paciente,fecha_ingreso,fecha_sintomas,
    mapping => { "message" => "%{fecha_actualizacion},%{id_registro},%{origen},%{sector},%{entidad_um},%{sexo},%{entidad_nac},%{entidad_res},
    %{municipio_res},%{tipo_paciente},%{fecha_ingreso},%{fecha_sintomas},%{fecha_def},%{intubado},%{neumonia},%{edad},%{nacionalidad},%{embarazo},
    %{habla_lengua_indig},%{indigena},%{diabetes},%{epoc},%{asma},%{inmusupr},%{hipertension},%{otra_com},%{cardiovascular},%{obesidad},
    %{renal_cronica},%{tabaquismo},%{otro_caso},%{toma_muestra},%{resultado_lab},%{toma_muestra_antigeno},%{resultado_antigeno},
    %{clasificacion_final},%{migrante},%{pais_nacionalidad},%{pais_origen},%{uci}" }
  }
```

Figura 41. Definición de cada uno de los atributos en Logstash.

Ahora, como se muestra en la Figura 42, se definen los formatos específicos para fechas, en especial para la fecha de defunción y la de ingreso, a las cuales se les define el formato de "día/mes/año" y en el caso de la fecha de síntomas que será nuestro pivote, se define como tipo "timestamp", ya que de esta manera mediante la interfaz de Kibana le indicaremos al patrón de índice que muestre la gráfica de registros de acuerdo a los meses en que han presentado síntomas los pacientes.

```
mutate {
        gsub => ["fecha_def", "9999-99-99","01/01/99"]
}

date {
        # 31/03/20
        # dd/MM/yy HH:mm:ss,SSS
        match => [ "fecha_ingreso", "dd/MM/yy" ]
        target => "f_ingreso"
}

date {
        # 31/03/20
        # dd/MM/yy HH:mm:ss,SSS
        match => [ "fecha_def", "dd/MM/yy" ]
        target => "f_def"
}

date {
        # 31/03/20
        # dd/MM/yy HH:mm:ss,SSS
        match => [ "fecha_sintomas", "dd/MM/yy" ]
        target => "@timestamp"
}
```

Figura 42. Definición del formato específico para las fechas que se tienen como atributos.

También es importante, llevar a cabo la actualización del tipo de dato de cada atributo, en este caso, indicamos uno por uno todos los que son de tipo "Integer", no es necesario indicar los que son de tipo cadena, ya que el mismo servidor de ElasticSearch reconoce cuales son de tipo "String"; la definición de datos se muestra en la Figura 43.

```
mutate {
  convert   => {
                "origen" => "integer"
                "sector" => "integer"
                "entidad_um" => "integer"
                "sexo" => "integer"
                "entidad_nac" => "integer"
                "entidad_res" => "integer"
                "municipio_res" => "integer"
                "tipo_paciente" => "integer"
                "intubado" => "integer"
                "neumonia" => "integer"
                "edad" => "integer"
                "nacionalidad" => "integer"
                "embarazo" => "integer"
                "habla_lengua_indig" => "integer"
                "indigena" => "integer"
                "diabetes" => "integer"
                "epoc" => "integer"
                "asma" => "integer"
                "inmusupr" => "integer"
                "hipertension" => "integer"
                "otra_com" => "integer"
                "cardiovascular" => "integer"
                "obesidad" => "integer"
                "renal_cronica" => "integer"
                "tabaquismo" => "integer"
                "otro_caso" => "integer"
                "toma_muestra" => "integer"
                "resultado_lab" => "integer"
                "toma_muestra_antigeno" => "integer"
                "resultado_antigeno" => "integer"
                "clasificacion_final" => "integer"
                "migrante" => "integer"
                "pais_origen" => "integer"
                "uci" => "integer"
              }
   remove_field => [ "message", "fecha_def", "fecha_sintomas"]
}
```

Figura 43. Definición del tipo de dato por cada uno de los atributos.

Finalmente, se define el nuevo nombre del índice que será reconocido en el servidor de Kibana, el cual indicará todo el conjunto de datos que se tendrán que gestionar en el motor de búsqueda, en este caso se nombró como “bd-covid-act-comorbilidades”, de esta manera al ejecutar el script de logstash, se creará un índice por cada mes con el que se cuentan registros, que es desde enero 2020 a mayo 2021 (ver Figura 44).

```
utput {

 elasticsearch {
    hosts => ["localhost:9200"]
    index => "bd-covid-act-comorbilidades-%{+YYYY.MM}"
 }

 stdout { codec => rubydebug }
```

Figura 44. Salida al servidor de ElasticSearch con el nuevo nombre de la base de datos actualizada.

4.4 Fase 4: Descubrimiento de patrones en los datos

En esta cuarta y última fase de la metodología, se lleva a cabo el análisis de los datos. En la sección 4.4.1 se describe la preparación preliminar del conjunto de datos a analizar; en la sección 4.4.2 se explica todo el pre-procesamiento para la aplicación de técnicas de minería de datos y finalmente en la sección 4.4.3 se presenta la evaluación de la información, obteniendo nuevos valores sobre el conjunto de datos de entrenamiento para proveer estas tendencias.

Aplicando técnicas de minería de datos, se esperan obtener características descriptivas que nos proporcionen puntos clave referentes a la pandemia de COVID 19 en relación con los registros de pacientes y como les ha afectado esta enfermedad. Para la aplicación de estas técnicas se utiliza el software WEKA, ya que es el más adecuado para manipular la información, aplicar algoritmos y descubrir patrones en los datos.

De igual forma, mediante el análisis de la información que se lleva a cabo por cada uno de los registros y al tener como datos las fechas de síntomas, de ingreso y defunción de los pacientes, evaluamos nuestro conjunto de datos realizando las consultas necesarias en SQL para obtener el total de población por entidad federativa. Además se calcula el número de personas que lamentablemente han fallecido a causa de esta enfermedad para que con ello, se realicen los cálculos correspondientes y se obtenga la tasa de mortalidad y letalidad por cada uno de los estados de la república, en base a la información que está manejando la Secretaría de Salud en México.

4.4.1 Preparación del conjunto de datos

Para una mejor obtención de resultados y un buen manejo de la información, ha sido necesario realizar algunas modificaciones a la base de datos que contiene el total de registros de pacientes; inicialmente se ha requerido llevar a cabo una reagrupación de los diferentes tipos de sector a los cuales ha acudido cada uno de los pacientes, quedando los grupos de la siguiente manera como se muestra en la Tabla 2.

Tabla 2. Sectores por grupo para las nuevas muestras de pacientes.

ID Sector	Grupo de Sectores
1	CRUZ ROJA - DIF - ESTATAL - MUNICIPAL - UNIVERSITARIO
2	IMSS - IMSS BIENESTAR
3	ISSSTE
4	PEMEX - SEDENA - SEMAR
5	SSA
6	PRIVADO

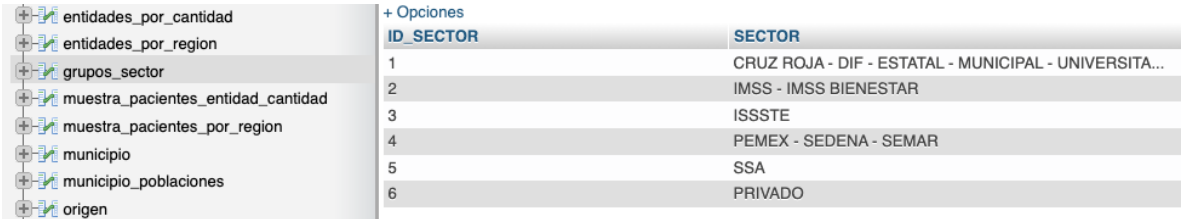
Por otra parte, también ha sido necesario realizar un agrupamiento de las entidades de residencia de acuerdo a la región geográfica a la que pertenecen, esto nos es útil para que al momento de analizar la información mediante WEKA y aplicar un algoritmo de clasificación como lo es el J48, el árbol de decisión que nos arroje esté construido de una forma mejor estructurada y por medio de esta agrupación el modelo predictivo que se obtenga nos arroje mejores resultados de acuerdo a la variable de clase que se esté evaluando y como influye en relación a los demás atributos. En la Tabla 3 se muestra como se definieron los nuevos grupos de entidades de acuerdo a la región a la que pertenecen.

Tabla 3. Grupos por entidades de residencia de acuerdo a la región geográfica a la que pertenecen.

ID Entidad	Región	Entidades
1	NOROESTE	BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CHIHUAHUA, SINALOA Y SONORA.
2	NORESTE	COAHUILA, DURANGO, NUEVO LEÓN, SAN LUIS POTOSÍ Y TAMAULIPAS.
3	OCCIDENTE	AGUASCALIENTES, COLIMA, GUANAJUATO, JALISCO, MICHOACÁN, NAYARIT, QUERÉTARO Y ZACATECAS.
4	CENTRO	CIUDAD DE MÉXICO, ESTADO DE MÉXICO, GUERRERO, HIDALGO, MORELOS, PUEBLA Y TLAXCALA.

5	SURESTE	CAMPECHE, CHIAPAS, OAXACA, QUINTANA ROO, TABASCO, VERACRUZ Y YUCATÁN.

Se crean las tablas correspondientes en la base de datos “covid19_act_comorbilidades” que es la que contiene todo el conjunto de datos, dichas tablas se muestran en la Figura 45 y Figura 46, de acuerdo a los grupos por sector y grupos de entidades por región respectivamente.



ID_SECTOR	SECTOR
1	CRUZ ROJA - DIF - ESTATAL - MUNICIPAL - UNIVERSITA...
2	IMSS - IMSS BIENESTAR
3	ISSSTE
4	PEMEX - SEDENA - SEMAR
5	SSA
6	PRIVADO

Figura 45. Tabla grupos_sector en la BD de COVID 19.

ID_ENTIDAD	REGION	ENTIDAD
1	NOROESTE	BAJA CALIFORNIA - BAJA CALIFORNIA SUR - CHIHUAHUA ...
2	NORESTE	COAHUILA - DURANGO - NUEVO LEÓN - SAN LUIS POTOSÍ ...
3	OCCIDENTE	AGUASCALIENTES - COLIMA - GUANAJUATO - JALISCO - M...
4	CENTRO	CIUDAD DE MÉXICO - ESTADO DE MÉXICO - GUERRERO - H...
5	SURESTE	CAMPECHE - CHIAPAS - OAXACA - QUINTANA ROO - TABAS...

Figura 46. Tabla entidades_por_region en la BD de COVID 19 con datos actualizados.

Una vez que se tienen las tablas con los nuevos grupos por sector y entidad, se deben realizar algunos scripts utilizando el lenguaje de programación Python para poder hacer una reclasificación de cada paciente registrado y asignar el nuevo ID Sector y ID Entidad al que pertenecen en su columna correspondiente y así insertar los nuevos conjuntos de datos.

En la Figura 47 se muestra el script con la función principal que se encarga de realizar una consulta a toda la información de la tabla “pacientes_clasificados”, en la cual se compara si el valor del ID_Sector que tiene asignado cada registro coincide con los valores de los nuevos grupos de sectores y si pertenece a dicho conjunto de sectores, entonces se le asigna el nuevo valor de ID. Todos estos nuevos registros se guardan en un nuevo archivo CSV, el cual deberá ser cargado con el script correspondiente que se ha utilizado desde un inicio para poder llevar a cabo la inserción de la información.

```
# Get Cursor
cur = conn.cursor()
cur.execute("SELECT * FROM pacientes_clasificados")
with open("datos_por_grupoSector.csv","w",encoding="utf8") as archivo:

    for (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION, FECHA_
    ID_PAIS_NACIONALIDAD, ID_PAIS_ORIGEN, UCI, DIAS_ATENDERSE, ESTADO_PACIENTE) in tqdm(cur):
        # Se procede a cambiar el sector de cada paciente de acuerdo a los nuevos grupos
        nuevo_sector=ID_SECTOR
        if nuevo_sector in ([1,2,3,7,13]):
            # pertenecen al grupo 1
            nuevo_sector=1
        elif nuevo_sector in ([4,5]):
            # pertenecen al grupo 2
            nuevo_sector=2
        elif nuevo_sector == 6:
            # pertenecen al grupo 3
            nuevo_sector=3
        elif nuevo_sector in ([8,10,11]):
            # pertenecen al grupo 4
            nuevo_sector=4
        elif nuevo_sector == 12:
            # pertenecen al grupo 5
            nuevo_sector=5
        elif nuevo_sector == 9:
            # pertenecen al grupo 6
            nuevo_sector=6

        archivo.write(f"{ID_REGISTRO},{ID_ORIGEN},{nuevo_sector},{ID_ENTIDAD_UM},{SEXO},{ID_ENTIDAD_NAC},{ID_ENTIDAD_RES},{ID_MUNICIPIO_RES},{TIPO_PACIENTE},
        SULTADO_ANTIGENO},{CLASIFICACION_FINAL},{MIGRANTE},{ID_PAIS_NACIONALIDAD},{ID_PAIS_ORIGEN},{UCI},{DIAS_ATENDERSE},{ESTADO_PACIENTE}\n")

        pass
```

Figura 47. Script en Python para asignar a los pacientes el nuevo grupo del sector al que pertenecen.

El script que se desarrolló para asignar el nuevo ID de la entidad de residencia de los pacientes, es similar al de los grupos por sector, en este caso se evalúa el atributo correspondiente a dicha entidad, si este pertenece al conjunto de datos específicos por región, entonces se le asigna un nuevo ID que va de 1 a 5.

Después se respalda cada uno de los registros de pacientes en un archivo CSV, en el cual a la hora que se van escribiendo los datos, en la columna correspondiente a la entidad de residencia, se le asignará el valor del nuevo ID obtenido. Al final, se lleva a cabo la inserción de los registros con el script para cargar el conjunto de datos dentro de la tabla "pacientes_region_entidad". La función principal se muestra en la Figura 48.

```python
# Get Cursor
cur = conn.cursor()
cur.execute("SELECT * FROM pacientes_grupo_sector")
with open("datos_region_entidad.csv","w",encoding="utf8") as archivo:

    for (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION, FEC
    ID_PAIS_NACIONALIDAD, ID_PAIS_ORIGEN, UCI, DIAS_ATENDERSE, ESTADO_PACIENTE) in tqdm(cur):
        # Se procede a cambiar la entidad de residencia de cada paciente de acuerdo a los nuevos grupos
        region_entidad=int(ID_ENTIDAD_RES)

        if region_entidad in ([2,3,8,25,26]):
            # pertenecen al grupo 1
            region_entidad=1
        elif region_entidad in ([5,10,19,24,28]):
            # pertenecen al grupo 2
            region_entidad=2
        elif region_entidad in ([1,6,11,14,16,18,22,32]):
            # pertenecen al grupo 3
            region_entidad=3
        elif region_entidad in ([9,15,12,13,17,21,29]):
            # pertenecen al grupo 4
            region_entidad=4
        elif region_entidad in ([4,7,20,23,27,30,31]):
            # pertenecen al grupo 5
            region_entidad=5

        # print(region_entidad)
        # input()
        archivo.write(f"{ID_REGISTRO},{ID_ORIGEN},{ID_SECTOR},{ID_ENTIDAD_UM},{SEXO},{ID_ENTIDAD_NAC},{region_entidad},{ID_MUNICIPIO_RES},{TIPO_PACIENTE},
        TADO_ANTIGENO},{CLASIFICACION_FINAL},{MIGRANTE},{ID_PAIS_NACIONALIDAD},{ID_PAIS_ORIGEN},{UCI},{DIAS_ATENDERSE},{ESTADO_PACIENTE}\n")

        pass
```

Figura 48. Función en Python que se encarga de asignar el nuevo ID de la entidad de residencia por cada paciente registrado.

Después, analizando el conjunto de datos, observamos que de acuerdo a la fecha de síntomas con la fecha de ingreso de los pacientes, se puede calcular el número de días que un paciente ha tardado en atenderse, con esto se puede comparar como ha influido el atenderse debidamente a tiempo con el estado actual del paciente, ya sea que se encuentre en remisión o bien ya haya fallecido.

Para llevar a cabo esta función se utiliza Python, en este lenguaje de programación se cuenta con la librería "Datetime" la cual nos permite trabajar adecuadamente con los registros de fechas que se tienen y saber exactamente cuantos días transcurren de una fecha a otra, es de manera similar a lo que se presentó en la sección 4.3.2, solo que en este caso se hacen las actualizaciones necesarias y las mejoras en cuanto a programación.

Para ello, se consulta la tabla "paciente_base" que contiene todos los registros almacenados y se hace una resta seleccionando el atributo de la fecha de ingreso menos la fecha de síntomas, con esto en el script se define la variable "días_transcurridos" como el resultado de cuantos días han tardado los pacientes en atenderse y acudir a una unidad médica de los diferentes sectores que se tienen.

Los registros de pacientes con el nuevo atributo de días transcurridos se respaldan en un archivo CSV para no saturar el servidor de MariaDB con tantas operaciones, después se procede a cargar dicho archivo a la tabla correspondiente utilizando el script que inserta todos los datos al servidor local. La función principal en Python se muestra en la Figura 49.

```
# Get Cursor
cur = conn.cursor()
cur.execute("SELECT * FROM paciente_base")
with open("importacion.csv","w",encoding="utf8") as archivo:

    for (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION, FECHA_INGRESO, FECHA_SINTOMAS,
        # Se transforma la fecha de ingreso
        fechaInicio=arreglarfechas(FECHA_INGRESO)
        fecha_ingreso = datetime.strptime(fechaInicio, '%d/%m/%Y')
        #Se transforma la fecha de defunsion
        fechaFin=arreglarfechas(FECHA_SINTOMAS)
        fecha_sintomas = datetime.strptime(fechaFin, '%d/%m/%Y')

        dias_t=(fecha_ingreso - fecha_sintomas)
        dias_transcurridos=int(dias_t.days)

        defuncion_estado='D'
        if FECHA_DEF == "9999-99-99":
            defuncion_estado='R'
            pass

        archivo.write(f"{ID_REGISTRO},{ID_ORIGEN},{ID_SECTOR},{ID_ENTIDAD_UM},{SEXO},{ID_ENTIDAD_NAC},{ID_ENTIDAD_RES},{ID_MUNICIPIO_RES},{TIPO_PACIENTE},{FECHA_ACTUALIZACION},{FECI
        \n")

        pass
```

Figura 49. Función que calcula los días que tardó el paciente en atenderse.

Ahora, tomando en cuenta la tabla de pacientes difuntos, podemos realizar un cálculo similar, para este caso calculan los días que ha estado el paciente hospitalizado antes de fallecer, por lo que con esto, se puede llevar a cabo un análisis del periodo de tiempo que ha durado este tipo de pacientes con la enfermedad y a su vez obtener características importantes de como influyen algunas comorbilidades o bien los días en recibir atención médica con su fallecimiento.

Para esto, se hace una resta entre la fecha de ingreso menos la fecha de defunción, por lo que se obtendrán los días que transcurrió con COVID 19 el paciente; de la misma manera que en la función anterior, con este script se respaldan todos los registros de pacientes en un nuevo archivo CSV pero aumentando el atributo de días de hospitalización y después se procede a cargar dicho archivo con el programa en Python que se encarga de insertar un conjunto de datos a la BD que es con el que se ha venido trabajando (ver Figura 50).

```
# Get Cursor
cur = conn.cursor()
cur.execute("SELECT * FROM pacientes_difuntos")
with open("pacientesDifuntos.csv","w",encoding="utf8") as archivo:

    for (ID_REGISTRO, ID_ORIGEN, ID_SECTOR, ID_ENTIDAD_UM, SEXO, ID_ENTIDAD_NAC, ID_ENTIDAD_RES, ID_MUNICIPIO_RES, TIPO_PACIENTE, FECHA_ACTUALIZACION, FECHA_INGRESO, FECHA_SINTOMAS
    ID_PAIS_NACIONALIDAD, ID_PAIS_ORIGEN, UCI) in tqdm(cur):
        # Se transforma la fecha de ingreso
        fechaInicio=arreglarfechas(FECHA_INGRESO)
        fecha_inicio = datetime.strptime(fechaInicio, '%d/%m/%Y')
        #Se transforma la fecha de defunsion
        fechaFin=arreglarfechas(FECHA_DEF)
        fecha_fin = datetime.strptime(fechaFin, '%d/%m/%Y')

        dias_t=(fecha_fin - fecha_inicio)
        dias_transcurridos=int(dias_t.days)

        defuncion_estado='D'
        if FECHA_DEF == "9999-99-99":
            defuncion_estado='R'
            pass

        archivo.write(f"{ID_REGISTRO},{ID_ORIGEN},{ID_SECTOR},{ID_ENTIDAD_UM},{SEXO},{ID_ENTIDAD_NAC},{ID_ENTIDAD_RES},{ID_MUNICIPIO_RES},{TIPO_PACIENTE},{FECHA_ACTUALIZACION},{FECI
        TADO_ANTIGENO},{CLASIFICACION_FINAL},{MIGRANTE},{ID_PAIS_NACIONALIDAD},{ID_PAIS_ORIGEN},{UCI},{dias_transcurridos},{defuncion_estado}\n")

        pass
```

Figura 50. Función que calcula los días de hospitalización de los pacientes difuntos.

Es importante añadir un atributo que es el estado del paciente, ya que nos servirá como variable de clase, la cual nos ayudará a hacer las predicciones necesarias en base a ese atributo aplicando algunos algoritmos de clasificación, en especial el J48 para obtener resultados de como se están comportando los datos. Esta variable se especificará por una sola letra donde R es igual a Recuperados y la letra D igual a Difuntos, de esta manera conoceremos el estado en el que se encuentra el paciente, únicamente se realiza una consulta al conjunto de datos y en un script de Python se analiza si el paciente tiene fecha de defunción se le asigna D y en caso contrario una R.

Esta función (ver Figura 51) se añade en el mismo script que calcula los días que tardó el paciente en atenderse, dentro del ciclo donde se están realizando las operaciones, se añade una extra que evalúe

si el paciente tiene fecha de defunción o no para así asignar el valor correspondiente el estado en el que se encuentra.

```
dias_t=(fecha_ingreso - fecha_sintomas)
dias_transcurridos=int(dias_t.days)

defuncion_estado='D'
if FECHA_DEF == "9999-99-99":
    defuncion_estado='R'
    pass
```

Figura 51. Función que asigna el estado del paciente a los datos.

4.4.2 Aplicación de técnicas de minería de datos

Para iniciar con la minería de datos, primero debemos obtener una muestra del total de pacientes, en este caso utilizaremos a los pacientes que se han insertado de acuerdo a las modificaciones por grupos de sector, así como el atributo entidad de residencia que está clasificado por el grupo de estadados a la región geográfica a la que pertenecen.

Esta muestra está considerada de acuerdo al tipo de clasificación final de los pacientes, se tienen registros con clasificación igual a1 que corresponde a los positivos por asociación clínica, 3 que son positivos confirmados por una prueba de laboratorio y clasificación 7 que corresponde a los pacientes que han dado negativo en la prueba. Esta muestra se toma desde la base de datos en MariaDB y se obtiene por medio de consultas en SQL como se muestra en la Figura 52. El total de registros de pacientes en esta muestra es de 30 265 datos.

```
--Creamos tabla para la muestra de pacientes por región
CREATE TABLE muestra_pacientes_por_region LIKE pacientes_region_entidad

--Insertar datos en la muestra por region
INSERT INTO muestra_pacientes_por_region SELECT * FROM pacientes_region_entidad
WHERE CLASIFICACION_FINAL = 1 AND FECHA_DEF != "9999-99-99"
ORDER BY RAND() LIMIT 6000

INSERT INTO muestra_pacientes_por_region SELECT * FROM pacientes_region_entidad
WHERE CLASIFICACION_FINAL = 1 AND FECHA_DEF = "9999-99-99"
ORDER BY RAND() LIMIT 6000

INSERT INTO muestra_pacientes_por_region SELECT * FROM pacientes_region_entidad
WHERE CLASIFICACION_FINAL = 3 AND FECHA_DEF != "9999-99-99"
ORDER BY RAND() LIMIT 6000

INSERT INTO muestra_pacientes_por_region SELECT * FROM pacientes_region_entidad
WHERE CLASIFICACION_FINAL = 3 AND FECHA_DEF = "9999-99-99"
ORDER BY RAND() LIMIT 6000

INSERT INTO muestra_pacientes_por_region SELECT * FROM pacientes_region_entidad
WHERE CLASIFICACION_FINAL = 7 AND FECHA_DEF != "9999-99-99"
ORDER BY RAND() LIMIT 6000

INSERT INTO muestra_pacientes_por_region SELECT * FROM pacientes_region_entidad
WHERE CLASIFICACION_FINAL = 7 AND FECHA_DEF = "9999-99-99"
ORDER BY RAND() LIMIT 6000
```

Figura 52. Consultas en SQL para obtener la muestra de pacientes para WEKA.

Una vez que se definió la muestra de datos para el análisis utilizamos WEKA, para ello es necesario realizar la conexión a la base de datos desde esta herramienta, se requiere tener instalado el driver de "mysql-connector-java" el cual permitirá la interacción de los datos con WEKA. Una vez que se establece la conexión se ejecuta una consulta a la tabla de la cual se desean obtener los datos a procesar.

En la Figura 53 se muestra el ejemplo de la conexión a la base de datos “covid19_act_comorbilidades” para obtener los registros de pacientes que están actualizados por grupos de sector y por la región de las entidades de residencia.

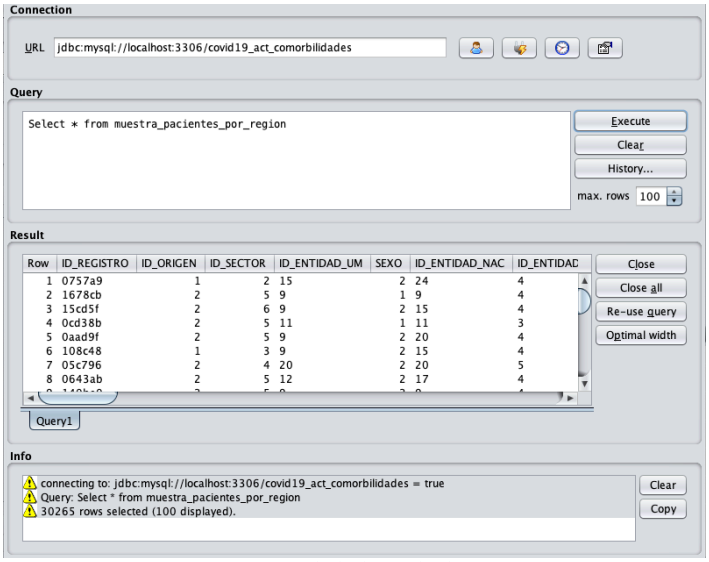


Figura 53. Conexión de la base de datos con WEKA.

Después de que se ha llevado a cabo la conexión con la BD, WEKA permitirá trabajar con la información y se mostrará de la siguiente manera (ver Figura 54):

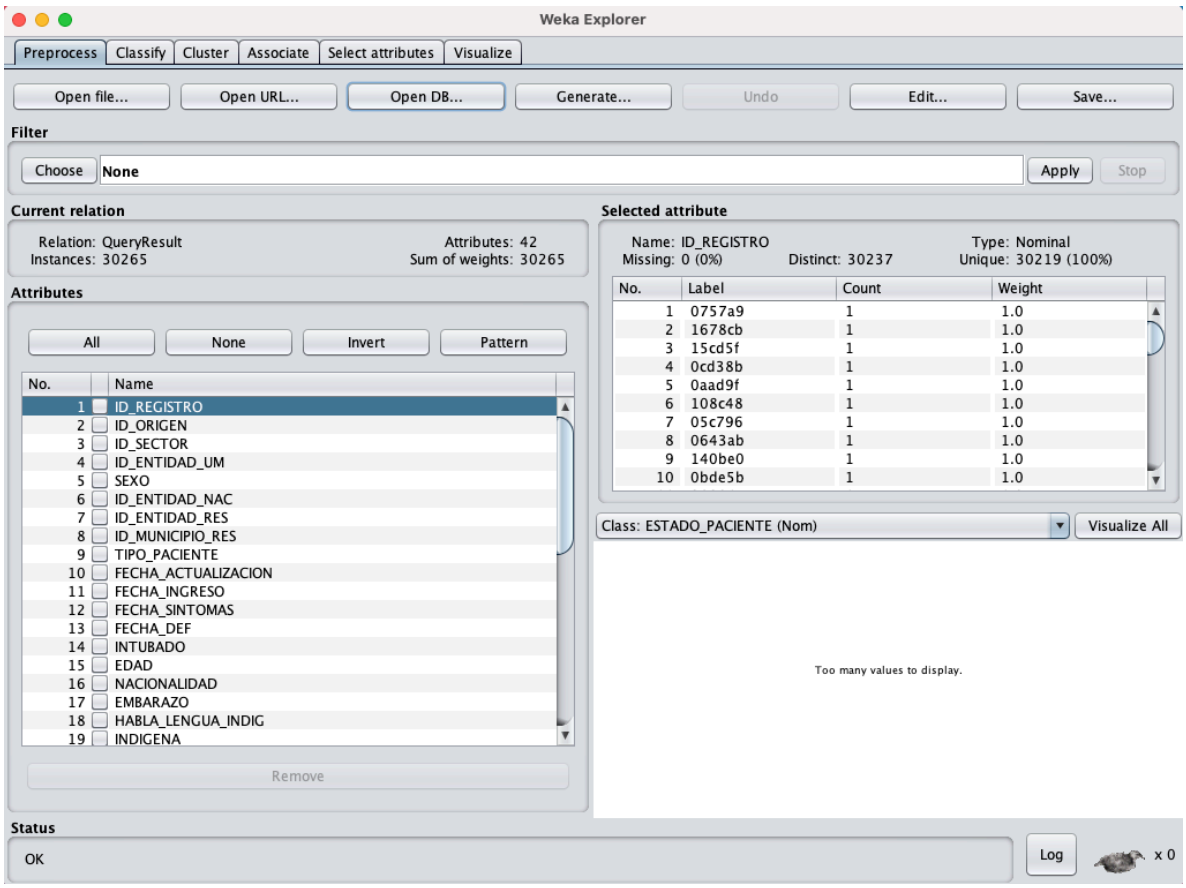


Figura 54. Datos presentados en WEKA para su análisis.

Del lado izquierdo se muestra la lista de atributos que contiene la tabla y del lado derecho al seleccionar un atributo se irán mostrando las instancias correspondientes a este.

Para poder trabajar con el conjunto de datos de entrenamiento, es necesario eliminar algunos atributos no relevantes para el análisis como lo son: id_registro, id_entidad_um, fecha_actualizacion, fecha_def (ya no la consideramos porque la variable de clase denota quienes son los pacientes difuntos), intubado, habla_lengua_indigena, indigena, otra_com, otro_caso, toma_muestra_ resultado_lab, toma_muestra_antigeno, resultado_antigeno, migrante, id_pais_nacionalidad, id_pais_origen y uci. Una vez que se seleccionaron estos atributos se les aplica la opción “Remove” y ahora el conjunto de datos queda de la siguiente manera como se muestra en la Figura 55.

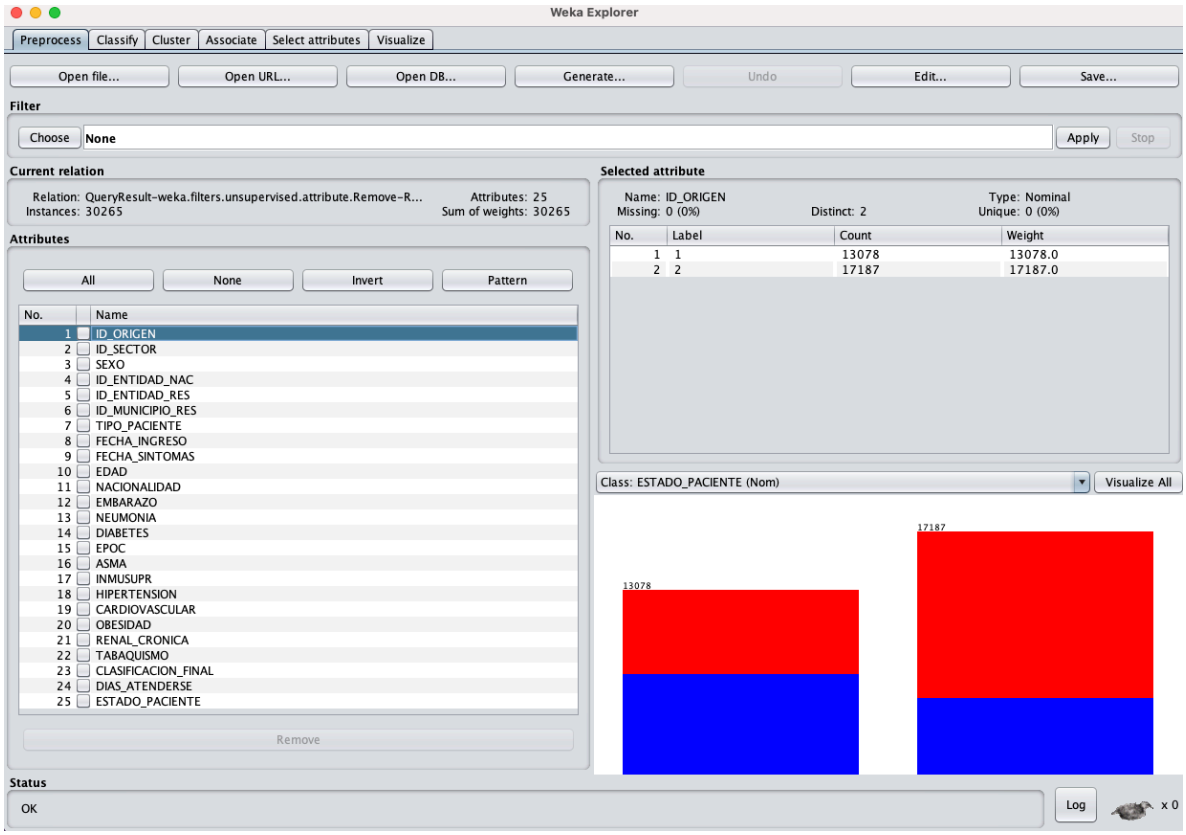


Figura 55. Conjunto de datos de entrenamiento sin atributos irrelevantes.

Después de que se ha limpiado el conjunto de datos, debemos convertir todos los atributos de tipo número a nominal, excepto la edad, este atributo lo trabajaremos de diferente manera. Primero seleccionamos el filtro no supervisado y por atributo llamado “NumericToNominal”, donde indicaremos cuales son los atributos que no queremos convertir, para este caso seleccionaré el atributo 10 correspondiente a la edad y en las opciones seleccionaré invertir selección, esto para que ese sea el único atributo que no se modificará (ver Figura 56).

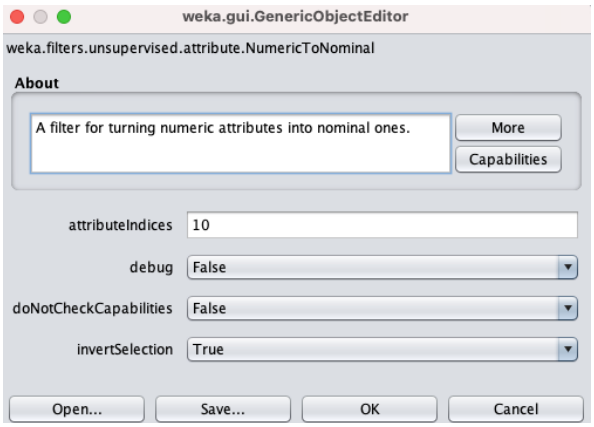


Figura 56. Opciones del filtro "NumericToNominal" en WEKA.

Como se había comentado anteriormente, el atributo “Edad”, se trabajará de diferente manera, para este caso se utilizará el filtro conocido como “Discretize”, el cual de manera similar al filtro “NumericToNominal”, convierte las instancias a tipo nominal, solo que en este caso nos permite seleccionar el número de grupos (bins) en el cual se clasificarán las instancias, en las opciones definimos 5 bins, ya que se requiere que la edad se clasifique en 5 grupos distintos los caules definirán los rangos de edad (ver Figura 57).

Figura 57. Opciones del filtro "Discretize" para convertir la edad a tipo nominal.

A partir de la discretización del atributo edad, se obtienen 5 grupos, donde los rangos por son: 0 – 24 años, 24 – 48 años, 48 – 72 años, 72 – 96 años y por último 96 años en adelante como se muestra en la Figura 58).

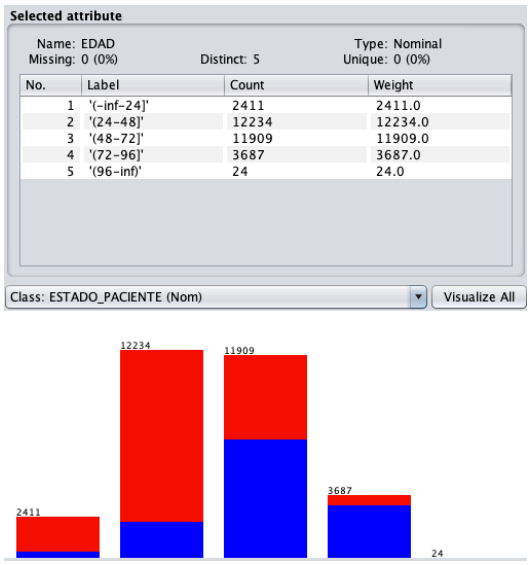


Figura 58. Grupos de edad formados al aplicar el filtro "Discretize" en WEKA.

Ya que se tiene el conjunto de datos de entrenamiento con los atributos relevantes, que sean de tipo nominal y la edad discretizada, utilizamos la variable de clase que obtuvimos en la sección anterior llamada "Estado_Paciente" para poder evaluar el modelo mediante un algoritmo de clasificación conocido como J48, de esta manera se obtiene como se están comportando los datos en relación a los pacientes recuperados y los difuntos. Los atributos que mejor predicen la variable de clase son: id_sector, tipo_paciente, edad, embarazo, diabetes, inmusupr y cardiovascular. En la Figura 59 se muestra este resultado mediante la opción del menú "Select attributes".

```
=== Attribute Selection on all input data ===

Search Method:
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 129
        Merit of best subset found:    0.447

Attribute Subset Evaluator (supervised, Class (nominal): 25 ESTADO_PACIENTE):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,7,10,12,14,17,19 : 7
                     ID_SECTOR
                     TIPO_PACIENTE
                     EDAD
                     EMBARAZO
                     DIABETES
                     INMUSUPR
                     CARDIOVASCULAR
```

Figura 59. Atributos que mejor predice la variable de clase Estado_Paciente.

Ahora, en el menú tenemos la pestaña "Classify" donde seleccionaremos el algoritmo J48, el cual está en la sección de árboles de decisión, ya que este método nos devuelve un árbol binario en el cual se presentan los resultados de la predicción.

En un primer caso al aplicar el algoritmo obtenemos un árbol donde inicialmente se divide de acuerdo al tipo de paciente (1 – ambulatorio, 2 – hospitalizado), para los pacientes de tipo ambulatorio se están considerado los que padecen de neumonía y los que no, en el caso que si padezcan neumonía se utilizan los rangos de edad para especificar la ganancia de pacientes recuperados en relación al conjunto de datos, se puede observar que de 48 a 72 años se predice mejor de acuerdo al id de la clasificación final, donde los positivos por prueba de laboratio se identifican de una mejor manera en

relación a la región por la entidad de residencia a la que pertenecen, mientras que los pacientes en el rango de 72 a 96 años de edad se identifican mejor de acuerdo al grupo de sector al que pertenecen (ver Figura 60).

```
TIPO_PACIENTE = 1
|   NEUMONIA = 1
|   |   EDAD = '(-inf-24]': R (48.0/4.0)
|   |   EDAD = '(24-48]': R (430.0/83.0)
|   |   EDAD = '(48-72]'
|   |   |   CLASIFICACION_FINAL = 1: R (114.0/23.0)
|   |   |   CLASIFICACION_FINAL = 3
|   |   |   |   ID_ENTIDAD_RES = 4: D (114.0/38.0)
|   |   |   |   ID_ENTIDAD_RES = 3: D (25.0/3.0)
|   |   |   |   ID_ENTIDAD_RES = 5: D (43.0/9.0)
|   |   |   |   ID_ENTIDAD_RES = 2: D (45.0/1.0)
|   |   |   |   ID_ENTIDAD_RES = 1: R (11.0/2.0)
|   |   |   CLASIFICACION_FINAL = 7: D (112.0/29.0)
|   |   EDAD = '(72-96]'
|   |   |   CLASIFICACION_FINAL = 1
|   |   |   |   ID_SECTOR = 1: R (0.0)
|   |   |   |   ID_SECTOR = 2: D (2.0/1.0)
|   |   |   |   ID_SECTOR = 3: D (3.0)
|   |   |   |   ID_SECTOR = 4: D (3.0/1.0)
|   |   |   |   ID_SECTOR = 5: R (8.0)
|   |   |   |   ID_SECTOR = 6: R (14.0/1.0)
|   |   |   CLASIFICACION_FINAL = 3: D (61.0/1.0)
|   |   |   CLASIFICACION_FINAL = 7: D (37.0/2.0)
|   |   EDAD = '(96-inf)': R (1.0)
|   NEUMONIA = 2
|   |   EDAD = '(-inf-24]': R (1778.0/11.0)
|   |   EDAD = '(24-48]': R (8741.0/112.0)
|   |   EDAD = '(48-72]': R (3714.0/301.0)
|   |   EDAD = '(72-96]'
|   |   |   ID_SECTOR = 1
|   |   |   |   ID_ORIGEN = 1: R (3.0)
|   |   |   |   ID_ORIGEN = 2: D (3.0/1.0)
|   |   |   ID_SECTOR = 2: D (102.0/28.0)
|   |   |   ID_SECTOR = 3: R (24.0/6.0)
|   |   |   ID_SECTOR = 4: R (33.0/15.0)
|   |   |   ID_SECTOR = 5: R (218.0/21.0)
|   |   |   ID_SECTOR = 6: R (29.0/6.0)
|   |   EDAD = '(96-inf)': R (2.0)
```

Figura 60. Primera parte del árbol, con el tipo de paciente igual a ambulatorios.

En el caso de los pacientes que son de tipo hospitalizado, la predicción se lleva a cabo de mejor manera en relación a los que han dado positivo y lo ha dictaminado una asociación clínica, además de dividirlos por el id de origen, que quiere decir si ha estado hospitalizado en una unidad de salud monitora de enfermedades respiratorias o no, para este caso los pacientes entre 48 y 72 años de edad se clasifican mejor de acuerdo al id de la región de entidades a la que pertenecen. Por otra parte, los pacientes que no se monitorean en este tipo de unidades de salud respiratorias, se consideran los que padecen neumonía y los pacientes que han acudido al sector privado son los que la variable de clase predice mejor cuando se clasifican por el id de entidad de residencia (ver Figura 61).

```
TIPO_PACIENTE = 2
|   CLASIFICACION_FINAL = 1
|   |   ID_ORIGEN = 1
|   |   |   EDAD = '(-inf-24]': R (34.0/1.0)
|   |   |   EDAD = '(24-48]': R (146.0/64.0)
|   |   |   EDAD = '(48-72]'
|   |   |   |   ID_ENTIDAD_RES = 4: D (318.0/85.0)
|   |   |   |   ID_ENTIDAD_RES = 3: R (9.0/3.0)
|   |   |   |   ID_ENTIDAD_RES = 5: D (41.0/11.0)
|   |   |   |   ID_ENTIDAD_RES = 2: D (8.0/3.0)
|   |   |   |   ID_ENTIDAD_RES = 1: R (16.0/6.0)
|   |   |   EDAD = '(72-96]': D (116.0/25.0)
|   |   |   EDAD = '(96-inf)': D (1.0)
|   |   ID_ORIGEN = 2
|   |   |   NEUMONIA = 1
|   |   |   |   ID_SECTOR = 1: R (17.0/8.0)
|   |   |   |   ID_SECTOR = 2: D (5.0)
|   |   |   |   ID_SECTOR = 3: D (76.0/34.0)
|   |   |   |   ID_SECTOR = 4: R (85.0/18.0)
|   |   |   |   ID_SECTOR = 5
|   |   |   |   |   ID_ENTIDAD_RES = 4: D (347.0/130.0)
|   |   |   |   |   ID_ENTIDAD_RES = 3: D (9.0/2.0)
|   |   |   |   |   ID_ENTIDAD_RES = 5: R (26.0/6.0)
|   |   |   |   |   ID_ENTIDAD_RES = 2: R (2.0)
|   |   |   |   |   ID_ENTIDAD_RES = 1: R (5.0)
|   |   |   |   ID_SECTOR = 6: R (312.0/83.0)
|   |   |   NEUMONIA = 2: R (651.0/85.0)
|   CLASIFICACION_FINAL = 3
|   |   EDAD = '(-inf-24]'
|   |   |   NEUMONIA = 1: D (43.0/12.0)
|   |   |   NEUMONIA = 2: R (44.0/11.0)
|   |   EDAD = '(24-48]': D (1287.0/411.0)
|   |   EDAD = '(48-72]': D (3719.0/541.0)
|   |   EDAD = '(72-96]': D (1355.0/101.0)
|   |   EDAD = '(96-inf)': D (6.0)
|   CLASIFICACION_FINAL = 7: D (5869.0/640.0)
```

Figura 61. Segunda parte del árbol, con el tipo de paciente igual a hospitalizados.

Partiendo del primer modelo (árbol obtenido del conjunto de datos de entrenamiento), podemos utilizar el factor de poda, el cual nos permitirá reducir el tamaño del árbol para hacer una nueva evaluación del conjunto de datos. El factor de poda siempre está definido en 0.25, ahora se modifica utilizando el valor de 0.025 para ver como se comporta el modelo y los resultados que arroja. En la Figura 62 se muestran las opciones del algoritmo J48 donde se modifica la opción llamada "confidenceFactor". Para que finalmente se muestren los resultados en la sección 5.4.

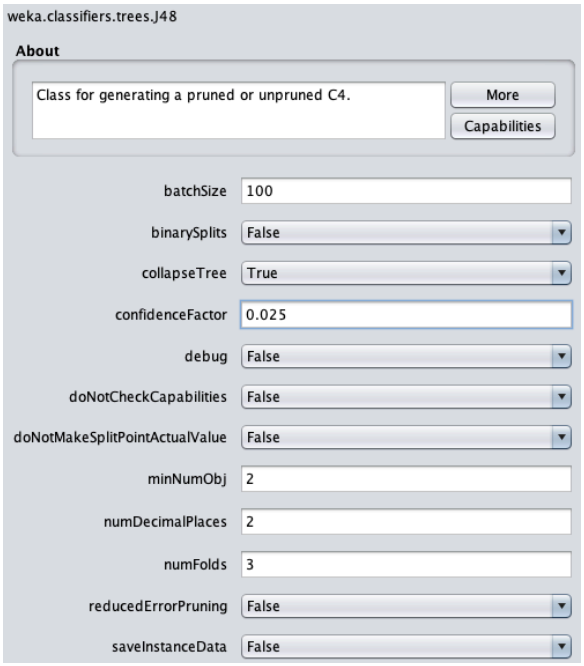


Figura 62. Opciones del algoritmo J48 en WEKA.

4.4.3 Evaluación de la información

Una vez que se ha analizado el conjunto de datos por medio de técnicas de aprendizaje automático como lo es la minería de datos, podemos realizar una evaluación de la información mediante la obtención de la tasa de mortalidad y letalidad, este cálculo es muy importante, ya que gracias al análisis de los datos que se tienen y todo el procesamiento de los mismos, se han obtenido nuevos atributos clave, los cuales nos proporcionan características importantes en relación a los pacientes que han padecido COVID-19.

Mediante el cálculo de datos derivados como los días transcurridos en atenderse, los días de hospitalización de los pacientes difuntos y el estado del paciente en conjunto con la información adicional que ya se tiene, nos es posible hacer esta aportación de indicar cual es la tasa de mortalidad y letalidad por cada uno de los estados de la República Mexicana, es decir, por cada entidad y cierto número de personas, se indicarán cuales son los valores de las tasas antes mencionadas, en relación a los datos proporcionados por la Secretaría de Salud de México.

Primero se requieren consultar los datos como el id de la entidad, el nombre del estado y su población total, ya que son la base para llevar a cabo las consultas a la base de datos, puesto que, se requiere realizar un conteo de todos los casos positivos a COVID-19 que corresponden a cada uno de los estados de la república. Para ello se crea la función en Python que se encarga de obtener el número de casos positivos registrados entre el rango de 1 a 32 que corresponden a los ID de las entidades, a su vez en el script se define la consulta SQL que se encarga de realizar dicho conteo, el resultado se guarda en una estructura de datos conocida como diccionario, en la cual Python organiza la información en forma de un arreglo de datos, la función se muestra en la Figura 63.

```
cur.execute("SELECT ID_ENTIDAD, ENTIDAD, POBLACION FROM entidad WHERE ID_ENTIDAD != 97 AND ID_ENTIDAD != 98 AND ID_ENTIDAD != 99 AND ID_ENTIDAD != 36")
with open("tasaMortalidad.csv","w",encoding="utf8") as archivo:
    cont=1
    diccionario_casos_entidad={}
    diccionario_difuntos_entidad={}
    for i in tqdm(range (32)):
        i=i+1
        #print(str(i)+"\n")
        cur2 = conn2.cursor()
        sql="SELECT COUNT(*) as Casos FROM paciente_base WHERE ID_ENTIDAD_RES = "+str(i)+";"
        #print(sql)
        cur2.execute(sql)
        resultado = (cur2.fetchone()[0])
        diccionario_casos_entidad[i]=resultado
        pass
```

Figura 63. Función para obtener el número de casos positivos por entidad.

Después, es necesario calcular el número de defunciones por entidad, para ello se crea una función similar en el mismo script de Python, solo que en este caso se realizará una consulta en SQL que cuente todos los pacientes que tengan asignada una fecha de defunción y el resultado se guardará en un diccionario distingo por el nombre “difuntos_entidad” como se muestra en la Figura 64.

```
for i in tqdm(range (32)):
    i=i+1
    #print(str(i)+"\n")
    cur2 = conn2.cursor()
    sql=f"SELECT COUNT(*) as Casos FROM paciente_base WHERE ID_ENTIDAD_RES = {i} AND FECHA_DEF != '9999-99-99';"
    #print(sql)
    cur2.execute(sql)
    resultado = (cur2.fetchone()[0])
    diccionario_difuntos_entidad[i]=resultado
    pass
```

Figura 64. Función para obtener el número de defunciones por entidad.

Habiéndose calculado ambos valores, lo que se tiene que hacer para calcular la tasa de mortalidad por cada entidad es una operación que consta de la división del número de defunciones entre el total de la población en esa entidad, este valor es unitario, es decir corresponde a la tasa de mortalidad por una sola persona; para tener un valor promedio, como por ejemplo cada mil personas, se realiza la división mencionada anteriormente y el resultado se multiplica por 1000, es decir que ese resultado será lo que equivale a cuantas personas mueren en dicha entidad por cada mil. Y finalmente ya que se tienen los resultados, estos mismos se guardan en un archivo CSV llamado “tasaMortalidad” para tener un respaldo y poder proporcionar la información (ver Figura 65).

```
print("Generando archivo de tasa de mortalidad\n")
for (id, entidad, poblacion) in tqdm(cur):
    casos_positivos=diccionario_casos_entidad[int(id)]
    casos_difuntos=diccionario_difuntos_entidad[int(id)]
    result_mortalidad_persona=(casos_difuntos/poblacion)
    mortalidad_promedio_1000=(casos_difuntos/poblacion)*1000

    archivo.write(f"{int(id)},{entidad},{poblacion},{casos_positivos},{casos_difuntos},{result_mortalidad_persona},{mortalidad_promedio_1000}\n")
    pass
```

Figura 65. Función que calcula la tasa de mortalidad de los pacientes por entidad.

Para calcular la tasa de letalidad del virus, el proceso es similar al anterior, lo único que cambia es la operación que se realiza para obtener el resultado, una vez que se tiene el número total de cosos positivos por entidad, así como el número de defunciones, ahora se procede a realizar la división del número de pacientes difuntos en determinada entidad entre el número de casos positivos, de esta manera se obtiene la letalidad unitaria; para ampliar el resultado a un valor promedio, multiplicamos lo que se obtuvo de la divisón por 100, lo que nos dará como resultado la tasa de letalidad por cada 100 personas, es decir, el resultado obtenido es la cantidad de pacientes que mueren por cada 100 personas que se encuentran enfermas de COVID-19 en determinada entidad. La función se muestra en la Figura 66 y de igual forma, los resultados se respalda en un archivo CSV llamado “tasaLetalidad”.

```
print("Generando archivo de tasa de mortalidad\n")
for (id, entidad, poblacion) in tqdm(cur):
    casos_positivos=diccionario_casos_entidad[int(id)]
    casos_difuntos=diccionario_difuntos_entidad[int(id)]

    tasa_letalidad=(casos_difuntos/casos_positivos)
    letalidad_promedio_100=(casos_difuntos/casos_positivos)*100
    archivo.write(f"{int(id)},{entidad},{poblacion},{casos_positivos},{casos_difuntos},{tasa_letalidad},{letalidad_promedio_100}\n")
    pass
```

Figura 66. Función que calcula la tasa de letalidad por cada 100 pacientes positivos por entidades.

Capítulo 5. Resultados

En esta sección se presentan los resultados experimentales obtenidos, en el caso de este proyecto de tesis se obtienen como resultado las herramientas de software desarrolladas a lo largo de toda la metodología propuesta. En la sección 5.1 se muestra el resultado de la implementación de la base de datos en el DBMS MariaDB, se presentan las tablas creadas para almacenar los registros de pacientes, así como su estructura y las actualizaciones realizadas. Por otra parte, en la sección 5.2 se presenta la BD creada en el motor de búsqueda ElasticSearch, el cual es adecuado para almacenar grandes volúmenes de información. Se muestra la gestión de índices para organizar la información almacenada, así como las gráficas con el total de registros en un rango de tiempo definido.

Finalmente, en la sección 5.3 se muestran los resultados que se han obtenido en base al análisis del conjunto de datos de entrenamiento, como son la tasa de mortalidad y letalidad de COVID-19, proporcionados por la Secretaría de Salud en México en conjunto con el CONACYT.

5.1 Base de datos en MariaDB

Con el desarrollo de este proyecto de tesis, se ha obtenido la implementación de una base de datos utilizando el sistema de gestión MariaDB, principalmente se ha utilizado el servidor de manera local, dicha BD se ha actualizado toda la información de registros de pacientes hasta el 31 de mayo de 2021 como se ha comentando en la sección 4.3.3 de la metodología.

En esta actualización de datos, se consideran las comorbilidades que padece el paciente en la tabla principal, además de tener los registros clasificados en la tabla “paciente_tiene_enfermedad”, es decir, en esta base de datos llamada “covid19_act_comorbilidades”, se cuenta con un mejor manejo de la información en cuanto a organización y estructura, teniendo un total de 24 tablas como se muestra en la Figura 67.

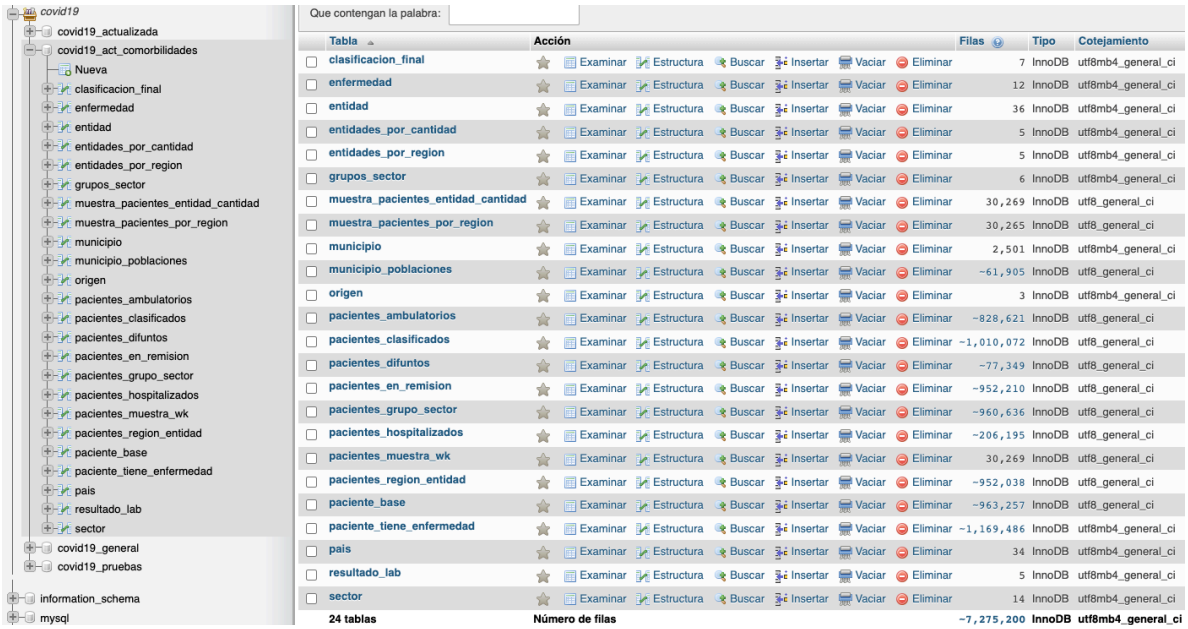


Figura 67. Tablas de la base de datos de COVID-19 actualizada.

De todo este conjunto de tablas se cuentan con algunas que son de las más relevantes, ya que, por ejemplo la tabla “paciente_base” (Figura 68) es la principal que cuenta con el total de registros que han sido proporcionados por la Secretaría de Salud en México hasta la fecha mencionada anteriormente, que cuentan 1 048 574 pacientes registrados en total. En la Tabla 4 se proporcionan los nombres con la cantidad de registros de las tablas con mayor relevancia en la base de datos de COVID-19.

ID_REGISTRO	ID_ORIGEN	ID_SECTOR	ID_ENTIDAD_UM	SEXO	ID_ENTIDAD_NAC	ID_ENTIDAD_RES	ID_MUNICIPIO_RES	TIPO_PACIENTE	FECHA_ACTUALIZACION	FECHA_INGRESO	FECHA_SINTOMAS	FECHA_DEF	INTUBADO	EDAD
z482b8	1	12	9	2	9	9	12	1	31/05/21	16/10/20	16/10/20	9999-99-99	97	41
z49a69	1	12	23	1	23	23	4	2	31/05/21	20/07/20	17/07/20	21/07/20	1	66
z23d9d	1	12	22	2	24	22	9	1	31/05/21	05/01/21	05/01/21	9999-99-99	97	29
z24953	1	12	9	1	9	9	10	1	31/05/21	15/10/20	15/10/20	9999-99-99	97	40
zz8e77	1	12	9	2	9	9	2	1	31/05/21	16/04/20	10/04/20	9999-99-99	97	34
z1b0d1	1	12	1	1	1	1	3	1	31/05/21	23/04/20	21/04/20	9999-99-99	97	48
z2d0c4	1	12	9	1	9	9	6	1	31/05/21	15/10/20	14/10/20	9999-99-99	97	60
z26b82	1	12	9	1	9	9	7	1	31/05/21	14/01/21	10/01/21	9999-99-99	97	20
z33a15	2	12	12	2	12	12	29	1	31/05/21	19/08/20	17/08/20	9999-99-99	97	47
z1b985	1	12	9	1	9	9	13	1	31/05/21	26/04/20	21/04/20	9999-99-99	97	40
z54974	1	12	9	2	9	9	12	1	31/05/21	10/10/20	01/10/20	9999-99-99	97	12
z51aa0	1	12	1	1	1	1	1	1	31/05/21	19/10/20	11/10/20	9999-99-99	97	33
zzcda2	1	12	2	1	2	2	4	1	31/05/21	03/03/20	01/03/20	9999-99-99	97	32
z3dc7b	1	12	9	2	9	9	11	1	31/05/21	16/05/20	16/05/20	9999-99-99	97	22
z4f06b	1	6	22	2	15	22	14	1	31/05/21	31/08/20	26/08/20	9999-99-99	97	54
z2770b	1	12	9	1	9	9	6	1	31/05/21	08/09/20	03/09/20	9999-99-99	97	26
z166d5	1	12	1	1	1	1	1	2	31/05/21	17/04/20	14/04/20	9999-99-99	2	32
z5702a	1	12	19	2	19	19	39	1	31/05/21	18/02/20	18/02/20	9999-99-99	97	55
zz2208	1	12	19	1	19	19	21	1	31/05/21	07/09/20	07/09/20	9999-99-99	97	51
z1ada4	1	12	19	2	19	19	39	1	31/05/21	13/07/20	13/07/20	9999-99-99	97	31
zz9079	1	12	9	1	9	9	7	1	31/05/21	07/10/20	06/10/20	9999-99-99	97	38
z388cd	1	12	29	2	15	21	117	2	31/05/21	01/07/20	26/06/20	9999-99-99	2	12
z4533d	1	12	29	1	21	21	117	2	31/05/21	02/07/20	28/06/20	9999-99-99	2	11
z21332	1	12	31	1	31	31	50	1	31/05/21	15/05/20	13/05/20	9999-99-99	97	19
z3f0f9	1	12	31	2	31	31	102	1	31/05/21	19/06/20	18/06/20	9999-99-99	97	60

Figura 68. Ejemplo de la tabla "paciente_base" con algunos de los atributos que corresponden a los datos de cada paciente.

Tabla 4. Cantidad de registros por cada una de las tablas de la BD.

Nombre de la tabla	Cantidad de registros
Paciente_base	1 048 574
Pacientes_ambulatorios	828 621
Pacientes_hospitalizados	206 195
Pacientes_en_remisión	952 210
Pacientes_difuntos	773 349
Pacientes_clasificados	1 048 574
Pacientes_grupo_sector	1 048 574
Pacientes_region_entidad	1 048 574
Paciente_tiene_enfermedad	1 169 486
Pacientes_muestra_wk	30 269
Muestra_pacientes_por_region	30 265

Las tablas llamadas “pacientes_ambulatorios” y “pacientes_hospitalizados” son una clasificación dependiendo el tipo de paciente, por otra parte, las tablas “pacientes_en_remision” y “pacientes_difuntos” se clasifican de acuerdo al atributo fecha de defunción, es decir, se compara si el paciente tiene tal fecha y si no la tiene se encuentra en remisión.

La tabla “pacientes_clasificados” es la tabla donde se agregó la variable de clase utilizada para las técnicas de minería de datos, la cual llamamos “estado_paciente”; en las tablas “pacientes_grupo_sector” y “pacientes_region_entidad” son las tablas donde actualizaron los ID del sector y de la entidad de residencia respectivamente, de acuerdo a los nuevos grupos de sectores y entidades por región que se formaron.

En la tabla “paciente_tiene_enfermedad” se insertaron los registros de todos los pacientes en relación a las comorbilidades que padecen, esta relación es de muchos a muchos; para que finalmente las tablas “pacientes_muestra_wk” y “muestra_pacientes_por_region” son una porción del total de registros donde se consideró el atributo “clasificación final” para seleccionar pacientes que han dado positivo

a COVID-19 por asociación clínica, por prueba de labotario y también los que han dado negativo, a su vez en la segunda tabla se han tomado en cuenta los pacientes donde el id de la entidad de residencia se ha actualizado de acuerdo a los grupos por región a la que pertenecen.

En la Figura 69 y Figura 70 se muestran las tablas de grupos_sector y entidades_por_region respectivamente, donde se observan como se definieron dichos grupos en base a los nombres de los sectores y entidades que se estaban manejando.

ID_SECTOR	SECTOR
1	CRUZ ROJA - DIF - ESTATAL - MUNICIPAL - UNIVERSITA...
2	IMSS - IMSS BIENESTAR
3	ISSSTE
4	PEMEX - SEDENA - SEMAR
5	SSA
6	PRIVADO

Figura 69. Representación de la tabla "grupos_sector".

ID_ENTIDAD	REGION	ENTIDAD
1	NOROESTE	BAJA CALIFORNIA - BAJA CALIFORNIA SUR - CHIHUAHUA ...
2	NORESTE	COAHUILA - DURANGO - NUEVO LEÓN - SAN LUIS POTOSÍ ...
3	OCCIDENTE	AGUASCALIENTES - COLIMA - GUANAJUATO - JALISCO - M...
4	CENTRO	CIUDAD DE MÉXICO - ESTADO DE MÉXICO - GUERRERO - H...
5	SURESTE	CAMPECHE - CHIAPAS - OAXACA - QUINTANA ROO - TABAS...

Figura 70. Representación de la tabla "entidades_por_region".

Por otra parte, las tablas generales de sector y entidad se muestran en la Figura 71 y Figura 72, donde se observan cada uno de estos datos de manera individual, para después poder formar los grupos mencionados en el párrafo anterior.

ID_SECTOR	SECTOR
1	CRUZ ROJA
2	DIF
3	ESTATAL
4	IMSS
5	IMSS-BIENESTAR
6	ISSSTE
7	MUNICIPAL
8	PEMEX
9	PRIVADA
10	SEDENA
11	SEMAR
12	SSA
13	UNIVERSITARIO
99	NO ESPECIFICADO

Figura 71. Tabla presentada en la BD de los sectores registrados.

ID_ENTIDAD	ENTIDAD	POBLACION
01	AGUASCALIENTES	1425607
02	BAJA CALIFORNIA	3769020
03	BAJA CALIFORNIA SUR	798447
04	CAMPECHE	928363
05	COAHUILA DE ZARAGOZA	3146771
06	COLIMA	731391
07	CHIAPAS	5543828
08	CHIHUAHUA	3741869
09	CIUDAD DE MÉXICO	9209944
10	DURANGO	1832650
11	GUANAJUATO	6166934
12	GUERRERO	3540685
13	HIDALGO	3082841
14	JALISCO	8348151
15	MÉXICO	16992418
16	MICHOACÁN DE OCAMPO	4748846
17	MORELOS	1971520
18	NAYARIT	1235456
19	NUEVO LEÓN	5784442
20	OAXACA	4132148
21	PUEBLA	6583278
22	QUERÉTARO	2368467
23	QUINTANA ROO	1857985
24	SAN LUIS POTOSÍ	2822255
25	SINALOA	3026943
26	SONORA	2944840
27	TABASCO	2402598
28	TAMAULIPAS	3527735
29	TLAXCALA	1342977
30	VERACRUZ DE IGNACIO DE LA LLAVE	8062579
31	YUCATÁN	2320898
32	ZACATECAS	1622138

Figura 72. Entidades de la República Mexicana registradas en la BD.

Para el caso de las enfermedades, cada una de ellas se representa en la tabla “enfermedades” donde se asigna un id seguido del nombre de la comorbilidad que puede padecer el paciente (ver Figura 73), mientras que en la tabla “paciente_tiene_enfermedad” es una relación de muchos a muchos, ya que, por cada registro de paciente de acuerdo a su id, se vincula con el id de la comorbilidad que padece, lo que nos da como resultado esa tabla (Figura 74) donde se especifica el id del paciente y todas las enfermedades que padece según el conjunto de datos.

ID_ENFERMEDAD	ENFERMEDAD
1	NEUMONÍA
2	DIABETES
3	EPOC
4	ASMA
5	INMUNOSUPRESIÓN
6	HIPERTENSIÓN
7	OTRA COMPLICACIÓN
8	CARDIOVASCULAR
9	OBESIDAD
10	RENAL CRÓNICA
11	TABAQUISMO
12	OTRO CASO

Figura 73. Representación de la tabla enfermedades en la BD.

ID_REGISTRO	ID_ENFERMEDAD
z49a69	1
z49a69	2
z49a69	6
z49a69	9
z49a69	12
z24953	12
zz8e77	3
zz8e77	4
z1b0d1	2
z1b0d1	9
z2d0c4	9
z33a15	2
z1b985	8
z1b985	12
z51aa0	4
z51aa0	12
zzcda2	12
z3dc7b	2
z3dc7b	3
z3dc7b	5
z3dc7b	6
z3dc7b	7
z3dc7b	8
z3dc7b	9
z3dc7b	11

Figura 74. Representación de la tabla "paciente_tiene_enfermedad" en la BD.

En la tabla “clasificacion_final” se tienen los indicadores que corresponden al id de la clasificación que se le ha dado a los pacientes, dependiendo si son positivos o negativos a COVID-19, para este caso hay siete clasificaciones diferentes que se muestran en la Figura 75 junto con el nombre de la clasificación y su descripción.

ID_CLASIFICACION_FINAL	CLASIFICACION	DESCRIPCION
1	CASO DE COVID-19 CONFIRMADO POR ASOCIACIÓN CLÍNICA EPIDEMIOLÓGICA	Confirmado por asociación aplica cuando el caso informó ser contacto de un positivo a COVID-19 (y este se encuentra registrado en el SISVER) y al caso no se le tomo muestra o la muestra resultó no válida.
2	CASO DE COVID-19 CONFIRMADO POR COMITÉ DE DICTAMINACIÓN	Confirmado por dictaminación solo aplica para defunciones bajo las siguientes condiciones: al caso no se le tomo muestra o sí se tomó muestra, pero la muestra resultó no válida.
3	CASO DE SARS-COV-2 CONFIRMADO POR LABORATORIO	Confirmado aplica cuando: el caso tiene muestra de laboratorio o prueba antigénica y resultó positiva a SARS-CoV-2, sin importar si el caso tienen asociación clínica epidemiológica.
4	INVÁLIDO POR LABORATORIO	Inválido aplica cuando el caso no tienen asociación clínico epidemiológica, ni dictaminación a COVID-19. Se le tomó muestra de laboratorio y esta resultó no válida.
5	NO REALIZADO POR LABORATORIO	No realizado aplica cuando el caso no tienen asociación clínico epidemiológica, ni dictaminación a COVID-19 y se le tomó muestra de laboratorio y esta no se procesó.
6	CASO SOSPECHOSO	Sospechoso aplica cuando: el caso no tienen asociación clínico epidemiológica, ni dictaminación a COVID-19 y no se le tomó muestra, o se le tomó muestra de laboratorio y está pendiente de resultado, sin importar otra condición.
7	NEGATIVO A SARS-COV-2 POR LABORATORIO	Negativo aplica cuando el caso: 1.- Se le tomó muestra de laboratorio y ésta resultó: negativa a SARS-COV-2 o positiva a cualquier otro virus respiratorio (Influenza, VSR, Bocavirus, otros) sin importar que este caso tenga asociación clínico epidemiológica o dictaminación a COVID-19. 2.- Se le tomó muestra antigénica que resultó negativa a SARS-COV-2 y al caso no se le tomó muestra de laboratorio ni se le confirmo por asosiación epidemiológica o por dictaminación clínica epidemiológica.

Figura 75. Información registrada en la tabla clasificación final para asignar a los pacientes.

Uno de los puntos importantes en la base de datos, ha sido el de agregar dos atributos importantes, como lo son los días que tardó el paciente en atenderse y la variable de clase que define el estado del paciente, letra R para recuperados y D para los difuntos, en la tabla “pacientes_clasificados” en las últimas columnas se pueden observar estos datos agregados que se calcularon mediante programas en Python.

Como se comentó en la sección 4.3.2 donde se hicieron modificaciones en la base de datos de MariaDB y también en la sección 4.4.1 donde se lleva a cabo la preparación de los datos para aplicar la minería, en esta sección se actualizó el procedimiento de lo que se realizó en la 4.3.2. En la Figura 76, se muestra la representación de la tabla mencionada con las dos columnas agregadas que corresponden a los valores que se calcularon para los días en atenderse y el estado del paciente.

ID_REGISTRO	ID_ENTIDAD_RES	FECHA_INGRESO	FECHA_SINTOMAS	FECHA_DEF	EDAD	CLASIFICACION_FINAL	DIAS_ATENDERSE	ESTADO_PACIENTE
z482b8	9	16/10/20	16/10/20	9999-99-99	41	1	0	R
z49a69	23	20/07/20	17/07/20	21/07/20	66	2	3	D
z23d9d	22	05/01/21	05/01/21	9999-99-99	29	6	0	R
z24953	9	15/10/20	15/10/20	9999-99-99	40	7	0	R
zz8e77	9	16/04/20	10/04/20	9999-99-99	34	6	6	R
z1b0d1	1	23/04/20	21/04/20	9999-99-99	48	5	2	R
z2d0c4	9	15/10/20	14/10/20	9999-99-99	60	7	1	R
z26b82	9	14/01/21	10/01/21	9999-99-99	20	7	4	R
z33a15	12	19/08/20	17/08/20	9999-99-99	47	3	2	R
z1b985	9	26/04/20	21/04/20	9999-99-99	40	7	5	R
z54974	9	10/10/20	01/10/20	9999-99-99	12	6	9	R
z51aa0	1	19/10/20	11/10/20	9999-99-99	33	7	8	R
zzcda2	2	03/03/20	01/03/20	9999-99-99	32	6	2	R
z3dc7b	9	16/05/20	16/05/20	9999-99-99	22	6	0	R
z4f06b	22	31/08/20	26/08/20	9999-99-99	54	3	5	R
z2770b	9	08/09/20	03/09/20	9999-99-99	26	3	5	R
z166d5	1	17/04/20	14/04/20	9999-99-99	32	3	3	R
z5702a	19	18/02/20	18/02/20	9999-99-99	55	6	0	R
zz2208	19	07/09/20	07/09/20	9999-99-99	51	7	0	R
z1ada4	19	13/07/20	13/07/20	9999-99-99	31	6	0	R
zz9079	9	07/10/20	06/10/20	9999-99-99	38	1	1	R
z388cd	21	01/07/20	26/06/20	9999-99-99	12	1	5	R
z4533d	21	02/07/20	28/06/20	9999-99-99	11	6	4	R
z21332	31	15/05/20	13/05/20	9999-99-99	19	7	2	R
z3f0f9	31	19/06/20	18/06/20	9999-99-99	60	7	1	R

Figura 76. Representación de los datos en la tabla "pacientes_clasificados" con los nuevos atributos agregados.

Por otra parte, de la clasificación hecha en la sección 4.4.1 para actualizar el id del sector de los pacientes a los nuevos id’s dependiendo el grupo de sectores, se presenta la Figura 77 como representación de la tabla “pacientes_grupos_sector” con los datos actualizados que fueron posibles modificar con el script de Python correspondiente; mientras que en la Figura 78 mostramos algunos valores de la tabla “pacientes_region_entidad” donde se puede apreciar el atributo id entidad de residencia con los nuevos valores del id de la región a la que pertenecen.

ID_REGISTRO	ID_ORIGEN	ID_SECTOR	ID_ENTIDAD_UM	SEXO	ID_ENTIDAD_NAC	ID_ENTIDAD_RES	ID_MUNICIPIO_RES	TIPO_PACIENTE
z482b8	1	5	9	2	9	9	12	1
z49a69	1	5	23	1	23	23	4	2
z23d9d	1	5	22	2	24	22	9	1
z24953	1	5	9	1	9	9	10	1
zz8e77	1	5	9	2	9	9	2	1
z1b0d1	1	5	1	1	1	1	3	1
z2d0c4	1	5	9	1	9	9	6	1
z26b82	1	5	9	1	9	9	7	1
z33a15	2	5	12	2	12	12	29	1
z1b985	1	5	9	1	9	9	13	1
z54974	1	5	9	2	9	9	12	1
z51aa0	1	5	1	1	1	1	1	1
zzcda2	1	5	2	1	2	2	4	1
z3dc7b	1	5	9	2	9	9	11	1
z4f06b	1	3	22	2	15	22	14	1
z2770b	1	5	9	1	9	9	6	1
z166d5	1	5	1	1	1	1	1	2
z5702a	1	5	19	2	19	19	39	1
zz2208	1	5	19	1	19	19	21	1
z1ada4	1	5	19	2	19	19	39	1
zz9079	1	5	9	1	9	9	7	1
z388cd	1	5	29	2	15	21	117	2
z4533d	1	5	29	1	21	21	117	2
z21332	1	5	31	1	31	31	50	1
z3f0f9	1	5	31	2	31	31	102	1

Figura 77. Actualización del id de sector para el conjunto de datos.

ID_REGISTRO	ID_ORIGEN	ID_SECTOR	ID_ENTIDAD_UM	SEXO	ID_ENTIDAD_NAC	ID_ENTIDAD_RES	ID_MUNICIPIO_RES	TIPO_PACIENTE
zzb8a2	1	5	21	2	21	4	114	2
z33ca2	1	5	29	2	29	4	31	1
1b04ab	1	5	9	1	9	4	14	1
09304e	1	5	31	2	31	5	96	1
16bcaf	1	5	7	1	7	5	65	1
169dce	1	5	31	2	31	5	102	1
1c4536	1	3	17	2	16	4	14	2
168645	1	1	7	1	7	5	101	1
08-feb-00	1	1	7	1	7	5	101	2
1d2168	1	5	10	2	10	2	32	1
1c3ca4	1	3	22	1	22	3	5	1
03e1c4	1	5	10	2	10	2	32	1
0e1a4b	1	5	7	1	7	5	59	1
0f948d	1	5	6	1	6	3	2	1
03eeb9	1	5	24	1	24	2	24	1
03d6f8	1	5	9	1	9	4	15	1
0fca35	1	5	31	2	31	5	89	1
02541b	1	5	22	2	22	3	14	1
0c5c89	1	5	27	1	27	5	16	1
0bf966	1	5	9	2	9	4	14	1
1b58cd	2	5	2	2	2	1	2	1
192632	1	5	1	1	1	3	1	1
84431	1	3	8	1	10	1	37	2
07f335	1	5	32	2	32	3	10	1
194a44	1	6	9	2	9	4	10	1

Figura 78. Actualización del id de la entidad de residencia por la región a la que pertenecen.

Finalmente, se dan a conocer los resultados de la muestra principal (ver Figura 79) que se tomó del total del conjunto de datos, siendo un total de 30 265 registros de pacientes, la cual se utilizó en la sección 4.4.2 para la aplicación de técnicas de minería de datos, esta muestra cuenta con el tipo de clasificación final 1, 3 y 7, de la cual ya se han explicado el significado de los valores en la sección mencionada con anterioridad.

En este conjunto de registros de pacientes con COVID-19, ya se tienen contemplados los nuevos atributos de días transcurridos en atenderse y el estado del paciente, además de tener actualizados el id del sector y de la entidad de residencia. Es importante recalcar que esta muestra para el conjunto de datos de entrenamiento fue procesada para poder obtener un mejor análisis y que fuera preciso para la predicción de datos en base a la variable de clase propuesta.

ID_REGISTRO	ID_ORIGEN	ID_SECTOR	ID_ENTIDAD_RES	TIPO_PACIENTE	FECHA_INGRESO	FECHA_SINTOMAS	FECHA_DEF	EDAD	CLASIFICACION_FINAL	DIAS_ATENDERSE	ESTADO_PACIENTE
0757a9	1	2	4	2	05/05/20	05/05/20	12/05/20	72	1	0	D
1678cb	2	5	4	2	19/04/20	15/04/20	19/04/20	22	1	4	D
15cd5f	2	6	4	2	14/07/20	11/07/20	15/07/20	76	1	3	D
0cd38b	2	5	3	2	22/07/20	17/07/20	24/07/20	78	1	5	D
0aad9f	2	5	4	2	01/05/20	21/04/20	10/05/20	62	1	10	D
108c48	1	3	4	2	23/05/20	15/05/20	29/05/20	72	1	8	D
05c796	2	4	5	2	10/07/20	30/06/20	13/07/20	70	1	10	D
0643ab	2	5	4	1	01/07/20	21/06/20	03/07/20	65	1	10	D
140be0	2	5	4	2	16/06/20	06/06/20	23/07/20	70	1	10	D
0bde5b	2	5	4	2	20/07/20	11/07/20	01/08/20	66	1	9	D
98901	2	3	3	2	21/06/20	19/06/20	22/06/20	54	1	2	D
17e05d	1	2	4	2	15/07/20	13/07/20	15/07/20	71	1	2	D
186597	2	6	4	1	20/04/20	17/04/20	20/04/20	42	1	3	D
040b1e	2	5	4	2	25/04/20	22/04/20	22/05/20	49	1	3	D
1d3ca5	1	5	4	2	11/06/20	02/06/20	13/06/20	57	1	9	D
08c2ac	1	3	4	2	02/07/20	01/07/20	30/07/20	70	1	1	D
0089c9	1	3	5	2	15/07/20	15/07/20	21/07/20	75	1	0	D
115bf0	2	5	4	2	03/06/20	26/05/20	03/06/20	44	1	8	D
17b6d2	2	4	4	1	11/05/20	09/05/20	12/05/20	32	1	2	D
04b848	2	3	4	2	14/05/20	01/05/20	09/06/20	77	1	13	D
009a86	1	2	4	2	04/06/20	03/06/20	14/06/20	70	1	1	D
05785f	1	5	4	2	06/06/20	01/06/20	07/06/20	55	1	5	D
06c52e	1	3	4	2	23/04/20	16/04/20	24/04/20	56	1	7	D
0202c4	1	2	4	2	06/07/20	06/07/20	07/07/20	43	1	0	D
1b94b4	2	2	4	2	11/06/20	11/06/20	28/06/20	52	1	0	D

Figura 79. Algunos datos de la muestra de pacientes por región para aplicar minería de datos.

5.2 Base de datos en ElasticSearch

Después de haber implementado toda la base de datos en MariaDB, es muy útil utilizar este motor de búsqueda basado en la nube, debido a que permite almacenar grandes volúmenes de información, es una gran opción para respaldar todo el conjunto de datos, así como para elegir la fecha de síntomas como clasificador principal y el mismo software se encargue de mostrar la gráfica de todos los pacientes registrados, dando opción de que se muestren por día, semana, mes o año, dependiendo del criterio que se tenga a la hora de consultar la información. En este software también se implementó una nueva base de datos con 1 048 574 pacientes registrados.

De la misma manera, como se comentó en la sección 4.3.1, al cargar todos los registros que se tienen en un archivo con formato CSV por medio del lenguaje Logstash, el servidor de ElasticSearch se comunica con la interfaz de Kibana la cual permite la interacción de la base de datos con el usuario y en esta se obtienen los nombres de los índices por cada mes referentes a la fecha de síntomas (ver Figura 80).

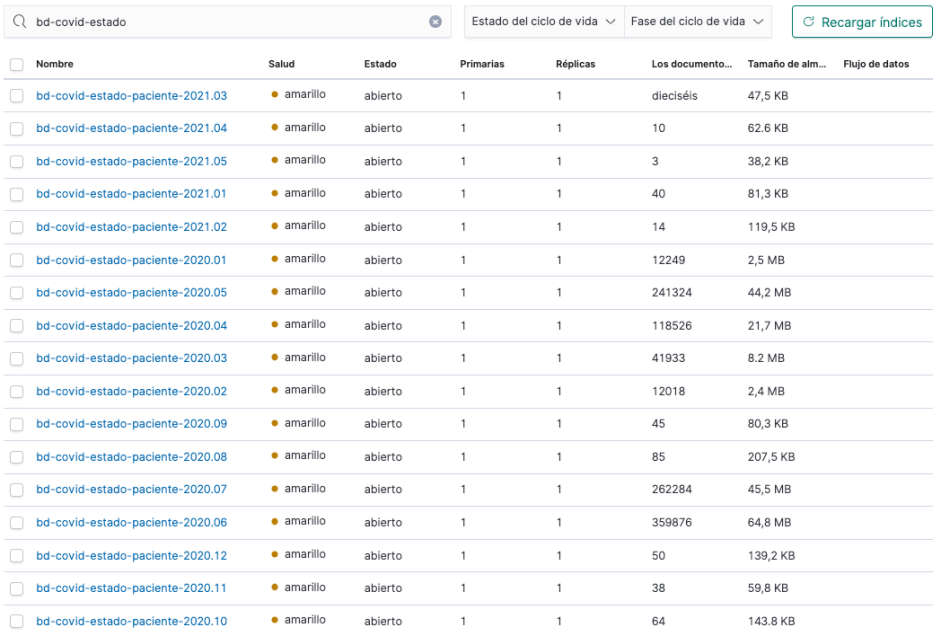


Figura 80. Gestión de índices creados para la BD de COVID-19 en ElasticSearch.

Una vez que se han identificado los índices correspondientes, se procede a crear el patrón de índice general, este se puede entender como el nombre que se le dará a la base de datos para que después toda la información se pueda mostrar en la ventana principal conocida como “Discovery”, la cual nos muestra las gráficas y datos que se explican en los siguientes párrafos, el patrón de índice se muestra como ejemplo en la Figura 81.

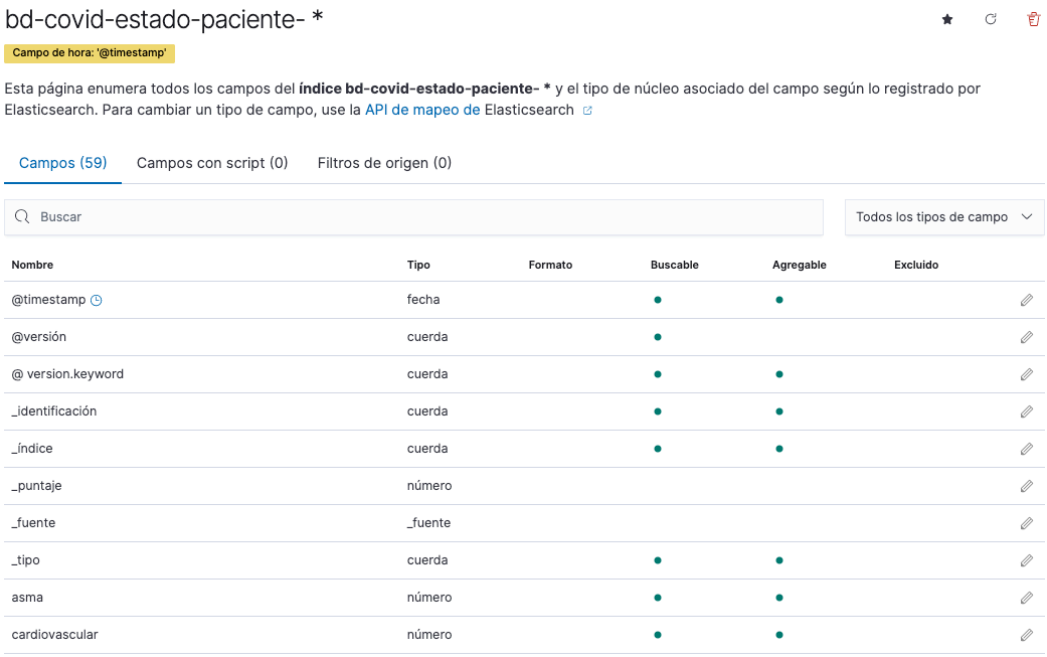


Figura 81. Patrón de índice creado que será el que contenga todo el conjunto de datos.

Como se muestra en la Figura 82, se tiene la gráfica de los registros de pacientes por semana, observando que los mayores picos de contagios están representados en la semana del 29 de junio y la del 6 de julio del año 2020. El intervalo de registros está dado desde la semana del 30 de diciembre de 2019 hasta la semana del 17 de mayo de 2021.

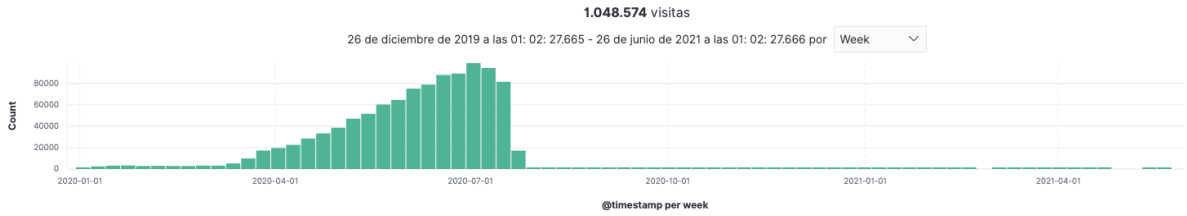


Figura 82. Gráfica por semana de los pacientes registrados con COVID-19 en ElasticSearch.

En la Tabla 5, se presentan el número total de registros por mes desde enero de 2020 hasta mayo de 2021, esta es una de las funcionalidades de ElasticSearch, que nos permite manejar los intervalos de tiempo para hacer representativos los datos y darlos a conocer de una manera mejor organizada. La gráfica correspondiente a este intervalo de meses se muestra en la Figura 83.

Tabla 5. Cantidad de pacientes registrados con COVID-19 por mes de Enero 2020 a Mayo 2021.

Mes	Cantidad de casos de COVID-19
Enero 2020	12 249
Febrero 2020	12 018
Marzo 2020	41 933
Abril 2020	128 526
Mayo 2020	241 324
Junio 2020	359 876
Julio 2020	262 283
Agosto 2020	85
Septiembre 2020	45
Octubre 2020	64
Noviembre 2020	38
Diciembre 2020	50
Enero 2021	40
Febrero 2021	14
Marzo 2021	16
Abril 2021	10
Mayo 2021	3

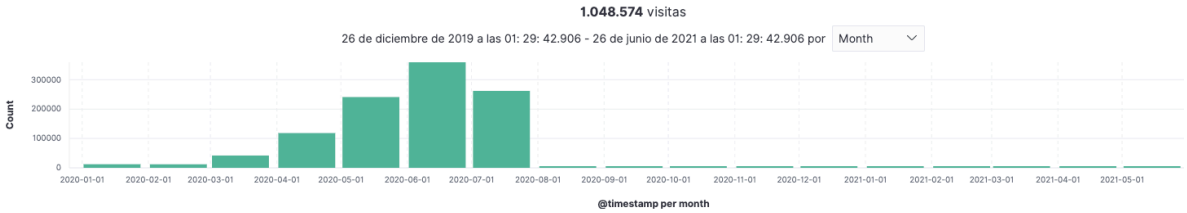


Figura 83. Gráfica de pacientes registrados por mes.

Dentro de la base de datos se muestran algunos registros de pacientes en la Figura 84, se puede observar que ElasticSearch no maneja el concepto de tablas, en este caso se le llama índices y se puede decir que es similar a una tabla debido a que se le asigna un nombre al índice para poder mostrar los datos y se clasifican por la variable definida de tipo "timestamp" que es la fecha de síntomas por parte de los pacientes.

Figura 84. Forma en que se presentan los datos registrados en la BD de ElasticSearch.

La estructura de uno de los registros dentro del servidor de ElasticSearch se presenta en la Figura 85, donde se observa del lado izquierdo el nombre de cada uno de los atributos seguido del valor asignado.

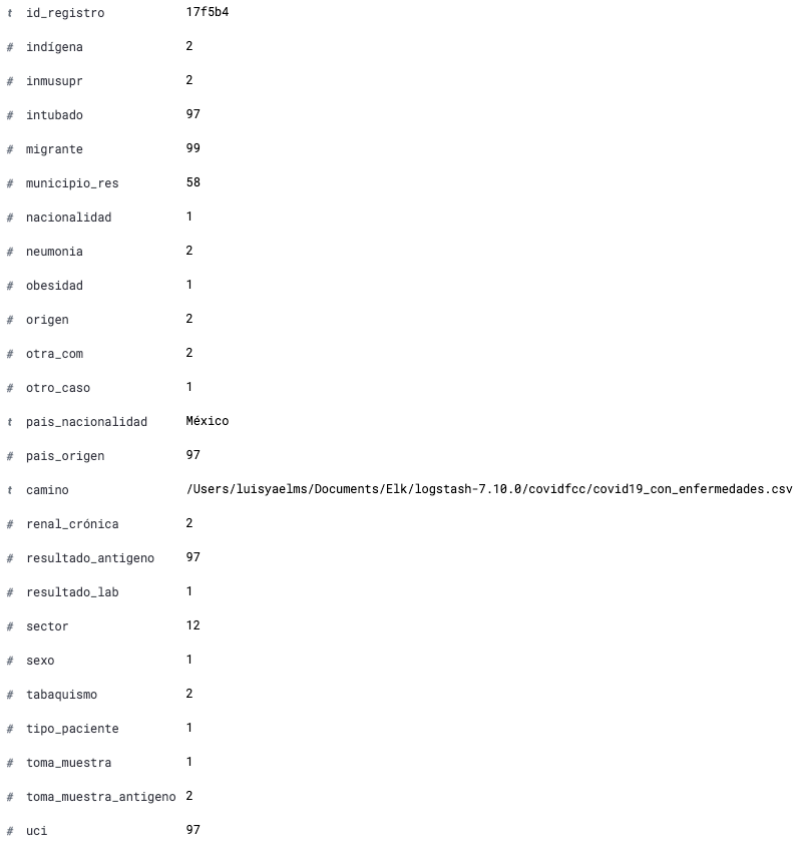


Figura 85. Estructura de uno de los registros que se insertan en el servidor de ElasticSearch.

Como se ha comentado anteriormente, para poder insertar un registro a la base de datos en esta herramienta de software, utiliza el formato JSON, el cual es parecido a un tipo de arreglo de datos, para lograr la estructura presentada en la figura anterior, es necesario que antes el registro a insertar haya sido transformado a JSON para que así el servidor pueda distinguir los atributos con su respectivo valor a asignar, el ejemplo de como se muestra un registro con este formato se presenta en la Figura 86.

```
{
  "_index": "bd-covid-act-comorbilidades-2021.05",
  "_type": "_doc",
  "_id": "52qWO3oBc4f2XDl-oXLZ",
  "_score": 1,
  "_source": {
    "edad": 48,
    "otra_com": 2,
    "cardiovascular": 2,
    "resultado_antigeno": 97,
    "origen": 2,
    "id_registro": "17f5b4",
    "f_def": "1999-01-01T06:00:00.000Z",
    "pais_origen": 97,
    "uci": 97,
    "path": "/Users/luisyaelms/Documents/Elk/logstash-7.10.0/covidfcc/covid19_con_enfermedades.csv",
    "pais_nacionalidad": "México",
    "indigena": 2,
    "host": "MacBook-Pro-de-Luis-2.local",
    "@version": "1",
    "hipertension": 2,
    "f_ingreso": "2021-05-20T05:00:00.000Z",
    "tipo_paciente": 1,
    "asma": 2,
    "nacionalidad": 1,
    "toma_muestra": 1,
    "intubado": 97,
    "entidad_nac": 9,
    "toma_muestra_antigeno": 2,
    "migrante": 99,
    "fecha_actualizacion": "31/05/21",
    "resultado_lab": 1,
    "tabaquismo": 2,
    "sector": 12,
    "clasificacion_final": 3,
    "otro_caso": 1,
    "neumonia": 2,
    "entidad_res": 15,
    "fecha_ingreso": "20/05/21",
    "inmusupr": 2,
    "sexo": 1,
    "diabetes": 2,
    "epoc": 2,
    "obesidad": 1,
    "entidad_um": 9,
    "municipio_res": 58,
    "embarazo": 2,
    "habla_lengua_indig": 2,
    "renal_cronica": 2,
    "@timestamp": "2021-05-18T05:00:00.000Z"
  },
```


Figura 86. Formato JSON para uno de los registros que serán insertados en ElasticSearch.

Finalmente, con toda esta implementación se obtiene una base de datos almacenada en la nube en un motor de búsqueda basado en las nuevas tecnologías para BD, con lo que ahora permite llevar un análisis más a fondo de cada uno de los datos debido a que cuenta con varias herramientas de análisis como la aplicación de técnicas de aprendizaje automático desde la herramienta de software como se muestra en la Figura 87.

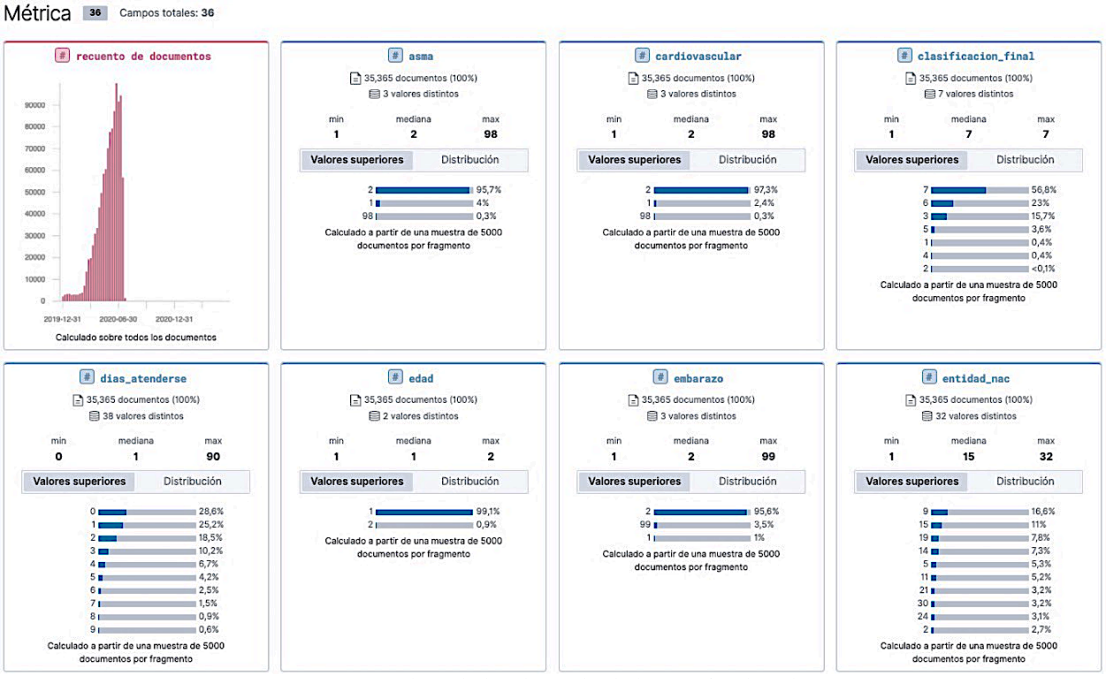


Figura 87. Análisis de los datos desde la interfaz de Kibana.

5.3 Resultados de la aplicación de técnicas de minería de datos

Aplicando el algoritmo J48 las nuevas modificaciones especificadas en la sección 4.4.2, se obtiene un árbol mejorado en el que se mantiene como nodo inicial la clasificación por tipo de paciente. En el caso del valor 1, correspondiente a los pacientes ambulatorios, se analizan en base a los que padecen neumionía o no, para los que la padecen se tiene que en el rango de 48 a 72 años de edad la variable de clase se evalúa de una mejor manera de acuerdo al id de la clasificación final que en los pacientes con edad de 72 a 96 años. En el caso de los pacientes ambulatorios que no padecen neumonía la variable de clase obtiene la probabilidad de pacientes recuperados en relación a dicha clasificación (ver Figura 88 con la primera parte del árbol).

```
TIPO_PACIENTE = 1
|   NEUMONIA = 1
|   |   EDAD = '(-inf-24]': R (48.0/4.0)
|   |   EDAD = '(24-48]': R (430.0/83.0)
|   |   EDAD = '(48-72]'
|   |   |   CLASIFICACION_FINAL = 1: R (114.0/23.0)
|   |   |   CLASIFICACION_FINAL = 3: D (238.0/60.0)
|   |   |   CLASIFICACION_FINAL = 7: D (112.0/29.0)
|   |   EDAD = '(72-96]'
|   |   |   CLASIFICACION_FINAL = 1: R (30.0/7.0)
|   |   |   CLASIFICACION_FINAL = 3: D (61.0/1.0)
|   |   |   CLASIFICACION_FINAL = 7: D (37.0/2.0)
|   |   EDAD = '(96-inf)': R (1.0)
|   NEUMONIA = 2: R (14647.0/548.0)
```

Figura 88. Primera parte del árbol obtenido con el factor de poda en 0.025.

En el caso de los pacientes hospitalizados (id igual a 2), se analiza el conjunto de datos de acuerdo a la clasificación final, donde los pacientes positivos por dictamen de asociación clínica, se dividen de acuerdo al id de origen, es decir si son monitoreados por una unidad médica de enfermedades respiratorias o no, en el caso que no pertenecen a las USMER (Unidades de Salud Monitoras de Enfermedad Respiratoria). Se analizan los registros de pacientes los cuales padecen neunomía o no; si la padecen, la predicción nos indica que los pacientes que se han atendido en el sector privado, la variable de clase devuelve una clasificación de acuerdo a la entidad de residencia a la que pertenecen y que probabilidad de pacientes difuntos y recuperados se obtiene en esos casos específicos. La Figura 89 muestra la rama del árbol de los pacientes hospitalizados.

```
TIPO_PACIENTE = 2
|   CLASIFICACION_FINAL = 1
|   |   ID_ORIGEN = 1
|   |   |   EDAD = '(-inf-24]': R (34.0/1.0)
|   |   |   EDAD = '(24-48]': R (146.0/64.0)
|   |   |   EDAD = '(48-72]': D (392.0/115.0)
|   |   |   EDAD = '(72-96]': D (116.0/25.0)
|   |   |   EDAD = '(96-inf)': D (1.0)
|   |   ID_ORIGEN = 2
|   |   |   NEUMONIA = 1
|   |   |   |   ID_SECTOR = 1: R (17.0/8.0)
|   |   |   |   ID_SECTOR = 2: D (5.0)
|   |   |   |   ID_SECTOR = 3: D (76.0/34.0)
|   |   |   |   ID_SECTOR = 4: R (85.0/18.0)
|   |   |   |   ID_SECTOR = 5
|   |   |   |   |   ID_ENTIDAD_RES = 4: D (347.0/130.0)
|   |   |   |   |   ID_ENTIDAD_RES = 3: D (9.0/2.0)
|   |   |   |   |   ID_ENTIDAD_RES = 5: R (26.0/6.0)
|   |   |   |   |   ID_ENTIDAD_RES = 2: R (2.0)
|   |   |   |   |   ID_ENTIDAD_RES = 1: R (5.0)
|   |   |   |   ID_SECTOR = 6: R (312.0/83.0)
|   |   |   NEUMONIA = 2: R (651.0/85.0)
|   CLASIFICACION_FINAL = 3: D (6454.0/1098.0)
|   CLASIFICACION_FINAL = 7: D (5869.0/640.0)
```

Figura 89. Segunda parte del árbol con un factor de poda en 0.025 y el tipo de pacientes es hospitalizados.

Finalmente se obtiene una clasificación con 89.7836% de instancias clasificadas correctamente, lo que nos indica que el conjunto de datos de entrenamiento utilizado ha generado un modelo satisfactorio como se muestra en la Figura 90.

```
Correctly Classified Instances        9213               89.5335 %
Incorrectly Classified Instances      1077               10.4665 %
Kappa statistic                          0.7876
Mean absolute error                      0.1718
Root mean squared error                  0.2962
Relative absolute error                 35.3932 %
Root relative squared error             60.0625 %
Total Number of Instances            10290

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.923    0.124    0.842      0.923    0.880      0.790    0.921     0.847     D
               0.876    0.077    0.941      0.876    0.907      0.790    0.921     0.926     R
Weighted Avg.  0.895    0.097    0.899      0.895    0.896      0.790    0.921     0.893
```

Figura 90. Resultados del modelo con la aplicación del algoritmo J48.

Con la aplicación del algoritmo J48 se obtienen las siguientes tendencias en base al árbol binario resultante del modelo analizado que se presenta en la Figura 91:

- Pacientes ambulatorios con neumonía y personas menores de 48 años, se recuperan el 2.3%. Mientras que los menores de 24 de años, solo se recupera el 0.2%.

- Para los pacientes ambulatorios con neumonía entre 48 y 72 años, que dieron negativo a la prueba de laboratorio, fallecieron el 1.75% más de los que se recuperaron.

- Los pacientes hospitalizados que dieron positivo por asociación clínica y están en una Unidad de Salud Monitoras de Enfermedades Respiratorias (USMER), mueren más los que están en el rango de edad de 48 a 72 años , así como de 72 a 96 años con el 2.54%.

- Por otra parte, para los pacientes que están fuera de USMER, tienen neumonía, además han sido atendidos por la Secretaría de Salud y Asistencia, se encontró un número mayor correspondiente a 1.73% de defunciones, en la región Centro de la República Mexicana, que corresponde a las entidades CDMX, Estado de México, Guerrero, Hidalgo, Morelos, Puebla y Tlaxcala.

- También se puede comprobar que los que han sido atendidos por el sector privado, tienen neumonía y están fuera de USMER, existe un mayor número de recuperados con el 1.56% en base a los resultados del árbol de decisión.

Se muestra el árbol binario que se obtiene como resultado de la aplicación de este algoritmo, donde el análisis correspondiente ha sido explicado anteriormente.

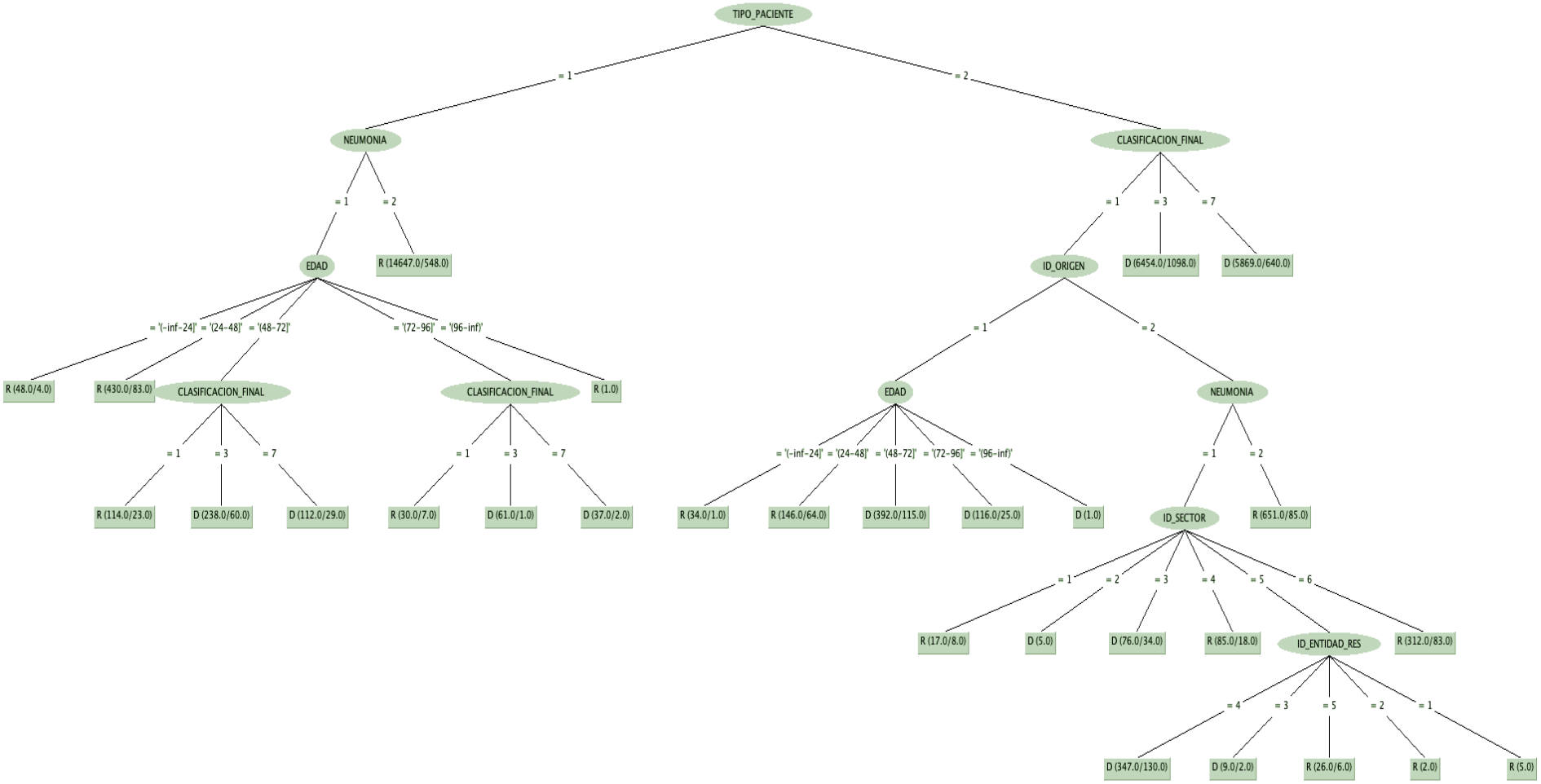


Figura 91. Árbol binario como resultado del algoritmo J48 al evaluar el conjunto de datos de entrenamiento.

En la gráfica de la Figura 92, se muestra la cantidad de pacientes recuperados y difuntos por rangos de edad, con esto podemos observar que las tendencias obtenidas por el árbol del algoritmo J48 son confiables, debido a que en el gráfico podemos visualizar en que rangos de edad hubo mas difuntos que recuperados y viceversa. Estos resultados son en base a la prueba de 30 265 pacientes.

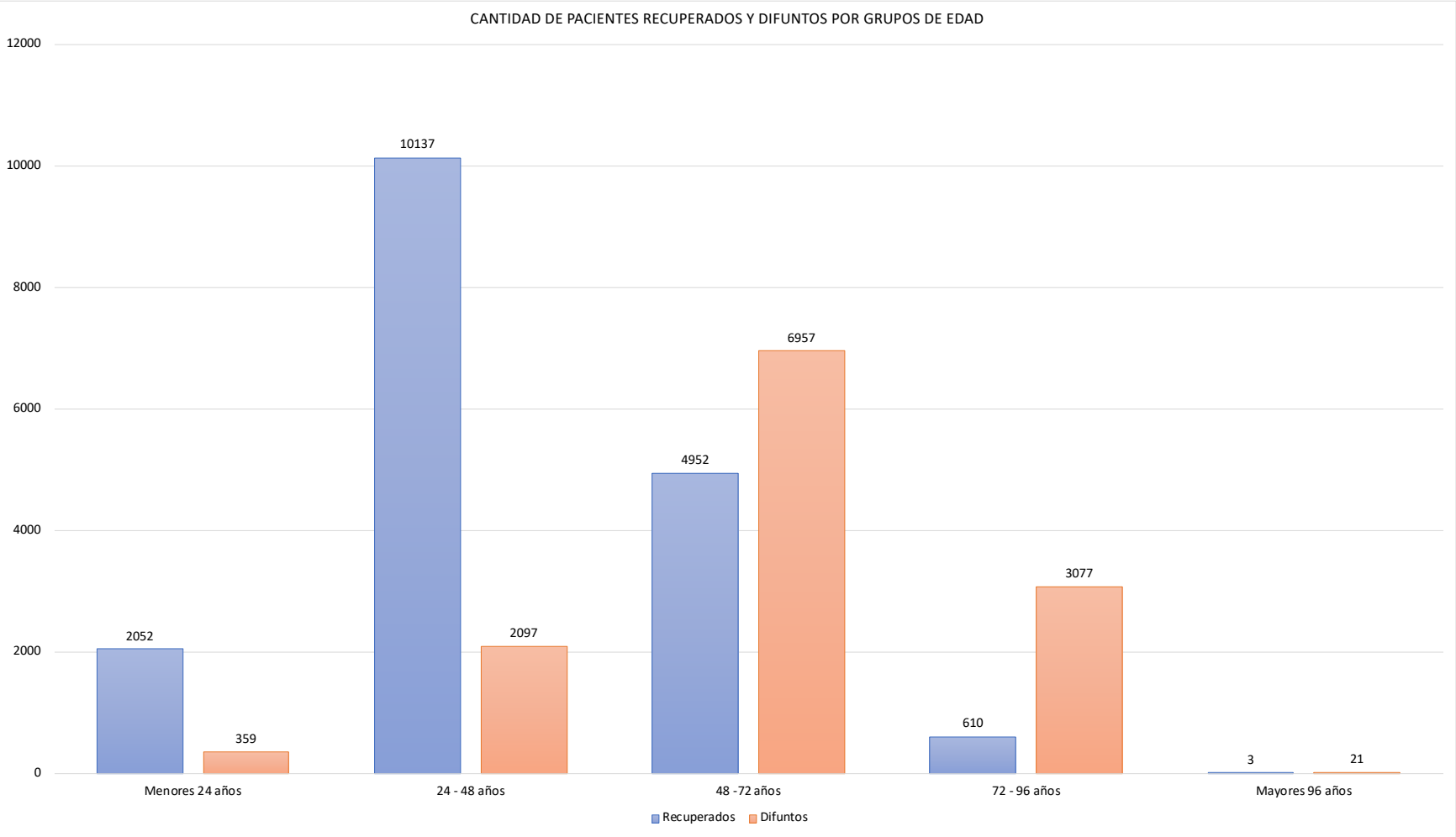


Figura 92. Gráfica por rangos de edad que muestra los pacientes recuperados y difuntos.

5.3.1 Análisis de parámetros

En la
Tabla 6, se obtienen como resultados del test, que las instancias correctamente clasificadas son 9213 frente a 1077 que fueron clasificadas de forma incorrecta.

Los resultados arrojados luego de la ejecución del algoritmo J48 mostraron que el coeficiente Kappa tuvo un valor de 0.7876, lo cual indica un nivel alto de concordancia entre los datos de prueba y los clasificados por el modelo, ya que cuanto más cercano a 1, mejor es la coincidencia de los datos.

El error absoluto medio tuvo un valor de 0.1718, lo cual demuestra que la diferencia entre las instancias de prueba y los valores reales son bajos, es decir, esto revela la eficiencia del modelo [61].

Dentro de los valores numéricos se encuentran el root mean squared error con un valor de 0.2962, el relative absolute error con un porcentaje de 35.39% y el root relative squared error con 60.06%. Todos estos valores se utilizaron para la predicción numérica en lugar de la clasificación; en la predicción numérica, estos errores reflejan una magnitud [62].

Tabla 6. Evaluación del test split.

Evaluation on Test Split		
Correctly Classified Instances	9213	89.53355%
Incorrectly Classified Instances	1077	10.4665%
Kappa statistic	0.7876	
Mean absolute error	0.1718	
Root mean squared error	0.2962	
Relative absolute error	35.3932%	
Root relative squared error	60.0625%	
Total Number of Instances	10290	

5.3.2 Detalle de precisión por clase

Estos resultados se muestran en la Tabla 7, como se puede observar, los verdaderos positivos (TP Rate) para la clase D – "defunción", superan el 92.3%, lo cual quiere decir que el árbol clasificó correctamente las instancias [63].

Los falsos positivos (FP Rate) tienen valores bajos, esto indica que el modelo pocas veces no clasificó correctamente las instancias.

Con respecto a la precisión, el porcentaje de acierto del modelo luego de hacer las clasificaciones en cada clase, se pueden observar valores mayores a 0.842, lo que demuestra que el modelo mide de la mejor manera las instancias correctamente reconocidas respecto al total de instancias predichas.

En los resultados de la cobertura (Recall), se puede observar que son altos, lo que significa que son favorables porque reconoce las instancias correctamente en cuanto a los términos reales. Es importante mencionar, el Recall y la Precisión están relacionadas entre sí, ya que cuando aumenta el Recall (la cobertura) disminuye la precisión o al contrario, si disminuye la cobertura aumenta la precisión. Se puede notar esta característica en la clase D, cuando el Recall es alto, la precisión tiene un valor menor, mientras que en la clase R, cuando la precisión es elevada, el Recall disminuye.

El F-Measure muestra la bondad del modelo, cuando el valor es más cercano a 1, mayor será la confiabilidad del modelo. Tal como se observa en la Tabla 7, el modelo demuestra ser confiable porque todos los valores de las clases se acercan a 1.

El ROC Area o área bajo la curva entre los verdaderos positivos (eje Y) y los falsos positivos (eje X), cuanto más cercano sea a 1 el test es visto como excelente [62].

En el análisis realizado para la muestra de pacientes con COVID-19 por región dependiendo la entidad a la que pertenecen, los resultados obtenidos son favorables, ya que la mayoría de los valores están cercanos a 1. Notablemente con estos valores se puede demostrar la confiabilidad del modelo.

Tabla 7. Detalles de precisión por clase.

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.923	0.124	0.842	0.923	0.880	0.790	0.921	0.847	D
0.876	0.077	0.941	0.876	0.907	0.790	0.921	0.926	R
0.895	0.097	0.899	0.895	0.896	0.790	0.921	0.893	

Nota: La última fila corresponde al peso promedio de valores calculados dependiendo la columna.

5.3.3 Matriz de confusión

Para la clase Defunción (D), se clasificaron correctamente 3958 instancias y 745 incorrectamente; para la clase Recuperados (R), se clasificaron 5255 instancias correctas y 332 incorrectas. En la Tabla 8, se observan los valores de la diagonal principal, los cuales son las instancias recuperadas de las opciones D y R.

Al notar que es mayor el número de instancias clasificadas correctamente, que el número de incorrectas, hace que se justifique lo explicado en las secciones 5.3.1 y 5.3.2, lo cual define que el modelo generado es confiable y de un acierto alto de clasificación.

Tabla 8. Matriz de confusión resultante de la evaluación del modelo con el algoritmo J48.

a	b	Clasificado como:
3958	332	**a = D**
745	5255	**b = R**

5.4 Obtención de la tasa de mortalidad y letalidad por entidad

A continuación se presentan los resultados obtenidos en base al procedimiento seguido en la sección 4.4.3 de la evaluación de la información con la que se ha estado trabajando, como es el caso de proporcionar el valor de la tasa de mortalidad y letalidad por cada entidad de la República Mexicana. En esa sección se explicó el procedimiento de como realizar dichos cálculos mediante programación en Python.
Primeramente, en la Tabla 9, se muestran las entidades con el valor total de su población por datos proporcionados por el INEGI del censo de población y vivienda 2020.

Tabla 9. Entidades con el total de población.

Nombre	Población total
Aguascalientes	1425607
Baja California	3769020
Baja California Sur	798447
Campeche	928363
Coahuila De Zaragoza	3146771
Colima	731391
Chiapas	5543828
Chihuahua	3741869
Ciudad De México	9209944
Durango	1832650
Guanajuato	6166934
Guerrero	3540685
Hidalgo	3082841
Jalisco	8348151
México	16992418
Michoacán De Ocampo	4748846
Morelos	1971520
Nayarit	1235456
Nuevo León	5784442
Oaxaca	4132148
Puebla	6583278
Querétaro	2368467
Quintana Roo	1857985
San Luis Potosí	2822255
Sinaloa	3026943
Sonora	2944840
Tabasco	2402598
Tamaulipas	3527735
Tlaxcala	1342977
Veracruz	8062579
Yucatán	2320898
Zacatecas	1622138

En la Tabla 10 se presentan datos como: el número de casos positivos por entidad y el número de defunciones, estos valores son en relación al conjunto de datos proporcionado por la Secretaría de Salud de México.

Tabla 10. Casos positivos y total de defunciones a causa de COVID-19 por entidad.

Nombre	Casos Positivos	Defunciones
Aguascalientes	13731	398
Baja California	37319	4033
Baja California Sur	10791	352
Campeche	10301	856
Coahuila De Zaragoza	36305	1367
Colima	4255	315
Chiapas	26356	1583
Chihuahua	14329	2112
Ciudad De México	192799	13691
Durango	12851	519
Guanajuato	51597	2382
Guerrero	18436	2120
Hidalgo	13057	1443
Jalisco	46162	2827

México	140010	15493	San Luis Potosí	24463	990
Michoacán De Ocampo	27262	1345	Sinaloa	26688	3469
Morelos	10628	1135	Sonora	31093	3204
Nayarit	7857	564	Tabasco	41141	2628
Nuevo León	52041	2187	Tamaulipas	38884	2304
Oaxaca	16522	1389	Tlaxcala	12778	1126
Puebla	42532	3818	Veracruz	36348	4816
Querétaro	9220	717	Yucatán	20684	1583
Quintana Roo	14666	1532	Zacatecas	7469	444

Los valores de la tasa de mortalidad por cada 1000 personas se presentan en la Tabla 11, estos resultados son obtenidos mediante los cálculos realizados con los scripts programados en Python como se ha comentando anteriormente.

Tabla 11. Tasa de mortalidad calculada por entidad federativa.

Nombre	**T.M. Por Cada 1000**	Michoacán De Ocampo	0.2832267
Aguascalientes	0.27917933	Morelos	0.57569794
Baja California	1.07003943	Nayarit	0.4565116
Baja California Sur	0.44085581	Nuevo León	0.37808314
Campeche	0.92205312	Oaxaca	0.33614478
Coahuila De Zaragoza	0.43441356	Puebla	0.57995424
Colima	0.43068619	Querétaro	0.30272746
Chiapas	0.28554277	Quintana Roo	0.82454918
Chihuahua	0.56442382	San Luis Potosí	0.35078333
Ciudad De México	1.48654541	Sinaloa	1.14604074
Durango	0.28319646	Sonora	1.08800478
Guanajuato	0.38625353	Tabasco	1.09381594
Guerrero	0.5987542	Tamaulipas	0.65311028
Hidalgo	0.46807474	Tlaxcala	0.8384358
Jalisco	0.33863786	Veracruz	0.59732748
México	0.91175959	Yucatán	0.68206358
		Zacatecas	0.27371284

Finalmente otro valor que se puede proporcionar es la tasa de letalidad, como ya se tienen valores como el total de población por entidad, así como el número de casos positivos y de defunciones, ahora se sigue el segundo procedimiento explicado en la sección 4.4.3 con el cual se calcula este valor mediante otro script programado en Python, obteniendo como resultado la Tabla 12; obteniéndose valores clave que se pueden proporcionar debido a que no se cuenta con este tipo de análisis en relación al conjunto datos de la Secretaría de Salud en México.

Tabla 12. Tasa de letalidad calculada por entidad.

Nombre	**T.L. Por Cada 100**
Aguascalientes	2.89855072
Baja California	10.8068276
Baja California Sur	3.26197757
Campeche	8.30987283
Coahuila De Zaragoza	3.76532158
Colima	7.40305523
Chiapas	6.00622249
Chihuahua	14.7393398
Ciudad De México	7.10117791
Durango	4.03859622
Guanajuato	4.61654747
Guerrero	11.4992406
Hidalgo	11.0515432
Jalisco	6.12408475
México	11.0656382
Michoacán De Ocampo	4.93360722
Morelos	10.6793376
Nayarit	7.17831233
Nuevo León	4.20245576
Oaxaca	8.40697252
Puebla	8.97677043
Querétaro	7.77657267
Quintana Roo	10.4459294
San Luis Potosí	4.04692801
Sinaloa	12.9983513
Sonora	10.3045702
Tabasco	6.38778834
Tamaulipas	5.92531633
Tlaxcala	8.81202066
Veracruz	13.2496974
Yucatán	7.65325856
Zacatecas	5.94457089

La siguiente gráfica de la Figura 93, muestra el total de casos positivos en conjunción con las defunciones que se han registrado en cada una de las entidades.

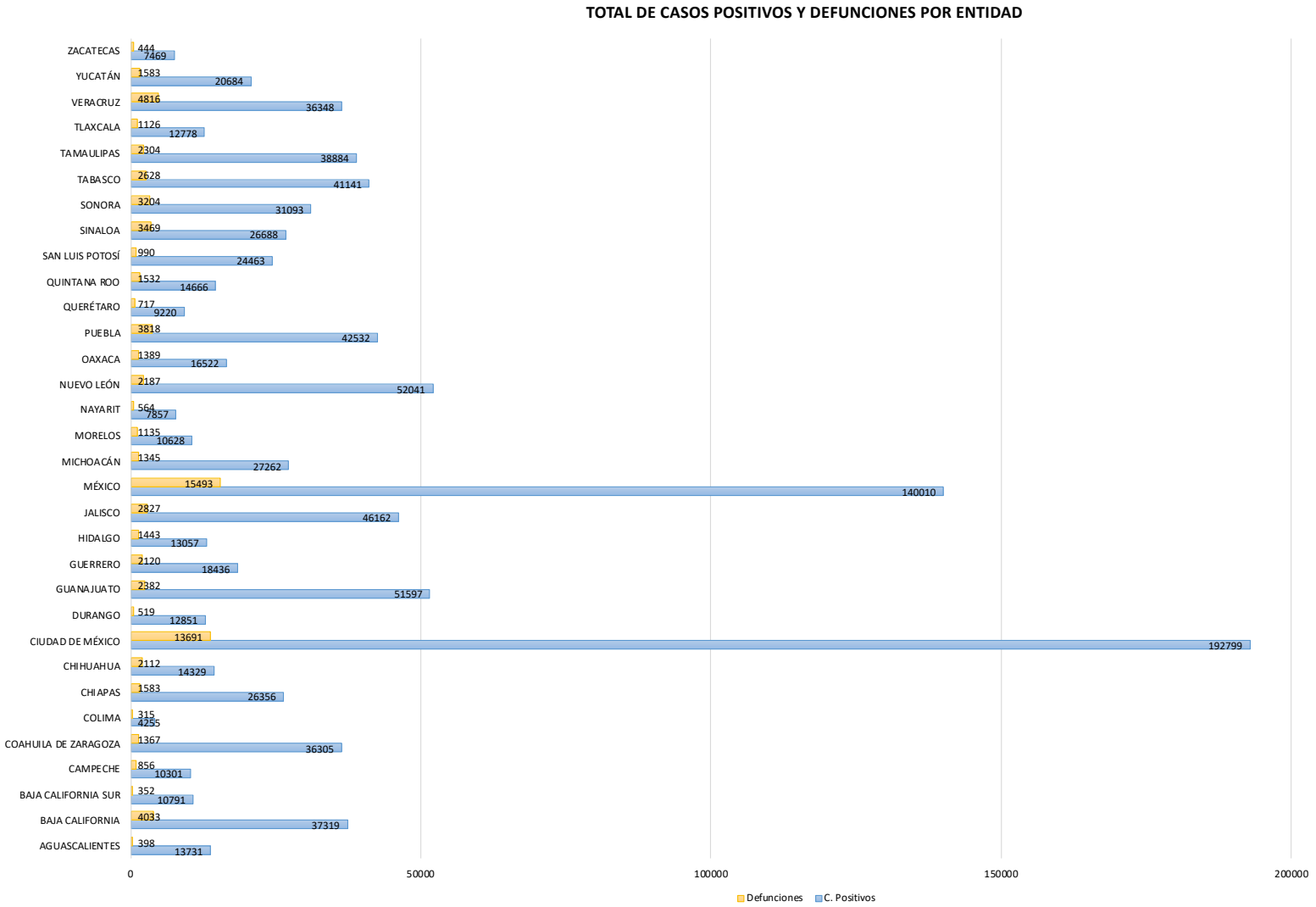


Figura 93. Gráfica que muestra el total de casos positivos y defunciones por entidad.

La siguiente gráfica de la Figura 94, muestra los valores del índice de mortalidad por cada una de las entidades de la república, donde se pueden observar los picos elevados en algunos estados, en los cuales se han presentado gran número de contagios.

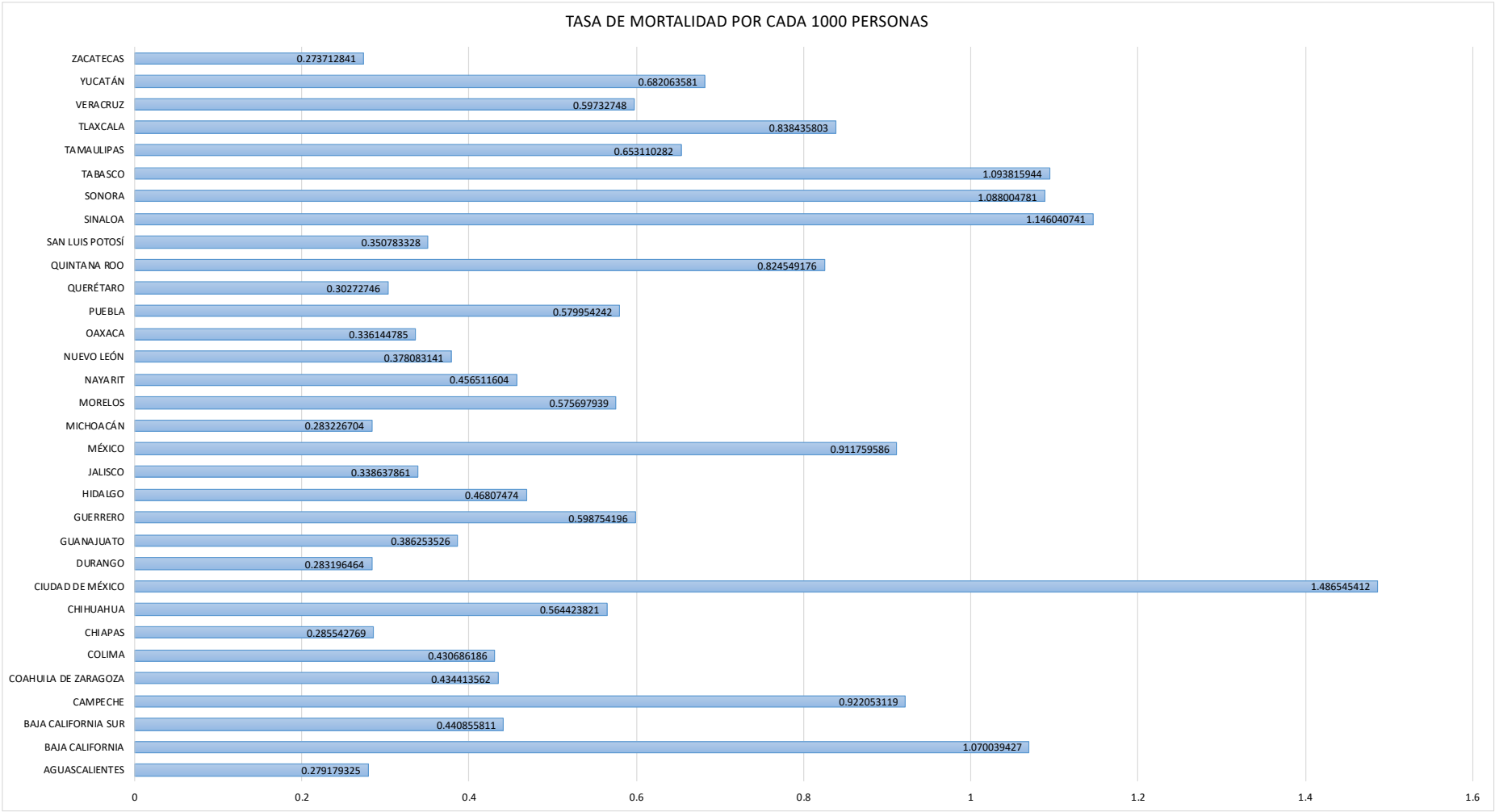


Figura 94. Gráfica que muestra la tasa de mortalidad por entidad.

La siguiente gráfica de la Figura 95, muestra la tasa de letalidad por cada 100 personas, se pueden observar en las barras correspondientes los valores pico por entidad, lo que quiere decir que en ese estado ha afectado de manera más significativa el COVID-19 en relación a los casos positivos con las defunciones.

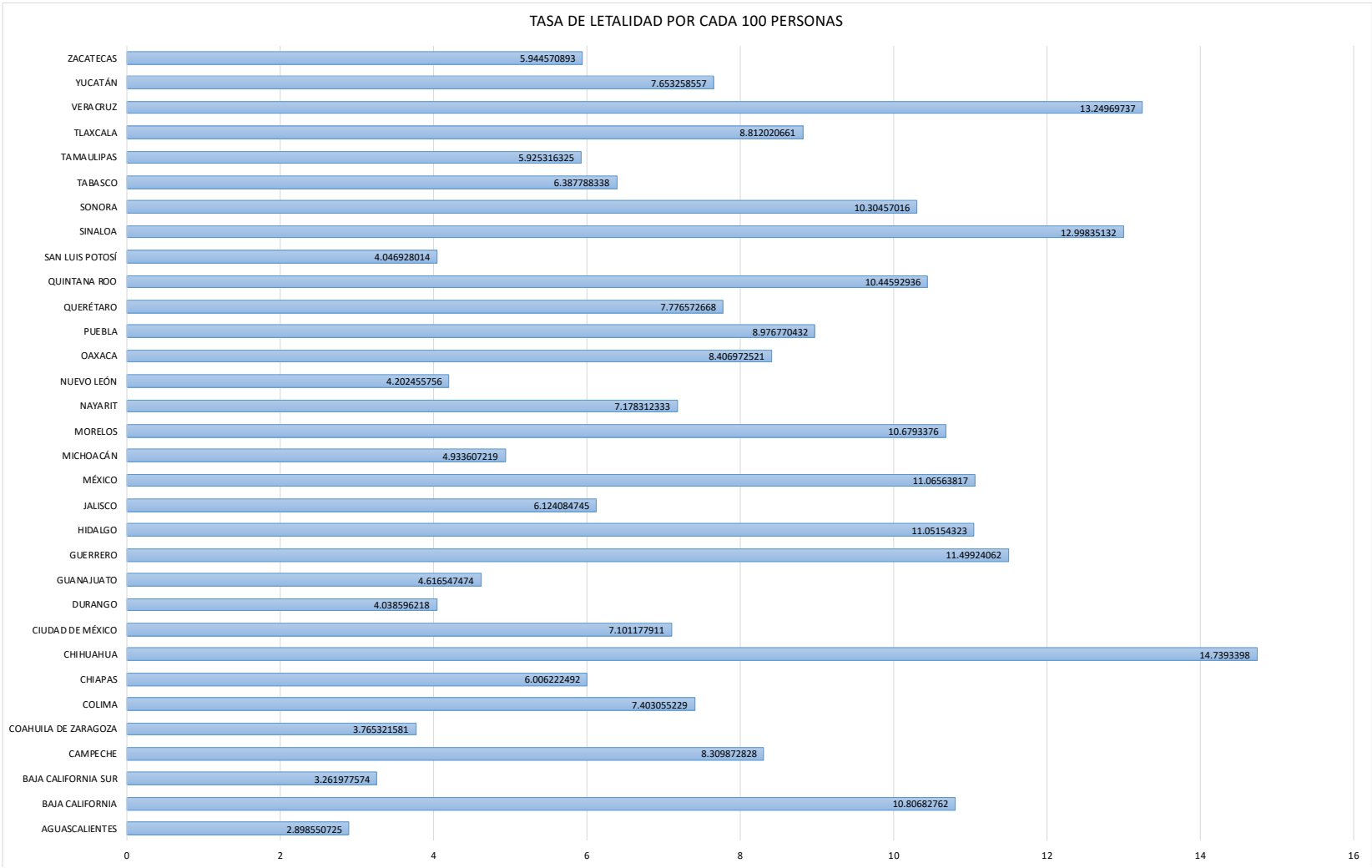


Figura 95. Gráfica que muestra la tasa de letalidad por entidad.

Capítulo 6. Conclusiones

Con el desarrollo de este trabajo de tesis, se llevó a cabo el proceso de búsqueda y recolección de información referente a la pandemia que se vive hoy en día a nivel mundial que es causada por COVID-19, se sigue una metología específica donde se aplican técnicas de Recuperación de Información así como de Big Data para la obtención de datos, principalmente utilizando la información proporcionada por la Secretaría de Salud del Gobierno de México. Cabe señalar que no se han encontrado trabajos relacionados donde manejen estos datos para un análisis y procesamiento más profundo con el cual puedan brindar nuevas aportaciones al área de salud pública en el país.

Inicialmente, se realizó la búsqueda en diferentes repositorios de información biomédica como: la Organización Mundial de la Salud, PubMed, Banco Mundial y la OPS, de los cuales se encontraban datos generales referentes a COVID-19 o bien de aplicaciones en el ámbito tecnológico, pero la mejor fuente de información que se optó para trabajar fue la de la Secretaría de Salud, debido a la gran cantidad de pacientes registrados y sus diferentes descriptores.

Para llevar a cabo el pre-procesamiento de los datos fue necesario realizar algunas modificaciones con programación en Python para poder darles una estructura homogénea que nos permitiera seguir con la etapa del diseño e implementanción de la base de datos.

Se construyó una base de datos en MariaDB en un entorno Apache en conjunción con el lenguaje de programación Python para insertar datos en formato ".csv".

Se desarrolló una segunda base de datos en un motor de búsqueda basado en la nube, ElasticSearch, este gestor brinda una amplia gama de herramientas, además de ser altamente escalable, lo que permitirá un mejor análisis en trabajos a futuro. Cabe recalcar, que en la actualidad no se cuenta con una herramienta de software que permita el manejo de la información de COVID-19 en México, haciendo uso de este tipo de tecnología.

El uso de MariaDB en conjunto con ElasticSearch, ha sido esencial, ya que toda la reestructuración de los datos se genera en el primer gestor que utiliza un entorno Apache, para que una vez teniendo los datos estructurados se proceda a almacenarlos en el motor de búsqueda y la información esté organizada.

Por otra parte, la aplicación de técnicas de minería de datos ha permitido predecir el comportamiento de la pandemia en base a atributos relevantes derivados: los días transcurridos sin atención médica y el estado (recuperado o difunto) en que se encuentra el paciente.

Finalmente, se obtienen tendencias como la tasa de mortalidad y letalidad de los pacientes con COVID-19 por cada entidad, realizando la programación correspondiente en Python y analizando algunos atributos de fechas que se tienen, estos valores se pueden proveer como información adicional del conjunto de datos proporcionado por la Secretaría de Salud.

Como trabajo a futuro para una continuación de este proyecto, se propone mejorar el conjunto de datos de entrenamiento para que con ello se apliquen otras técnicas de aprendizaje automático y llevar a cabo una mejor predicción de patrones en los datos. Además, también se pretende ampliar la investigación en repositorios de información biomédica donde se obtengan por ejemplo, expedientes médicos que brinden diferentes datos característicos de los pacientes enfermos para después poder desarrollar y proporcionar una herramienta que ayude a la prevención y diagnóstico de COVID-19.

Debido a que el conjunto de datos con que se cuenta no es suficiente para trabajar en el diagnóstico de la enfermedad, una propuesta alternativa para trabajar a futuro esta limitante, es utilizar el modelo SIR (Susceptible, Infectado y Recuperado). A grandes rasgos este modelo calcula el número teórico de personas infectadas con una enfermedad contagiosa en una población cerrada a lo largo del tiempo; mediante ecuaciones probabilísticas se realiza un mejor análisis de la información obteniendo así patrones precisos que pueden describir de mejor manera el comportamiento de los datos en relación a un paciente con COVID-19. Podría parecer interesante trabajar con este modelo para obtener características importantes que permitan deducir patrones para la prevención y diagnóstico de esta enfermedad y así poder contribuir con las autoridades sanitarias en la toma de decisiones referente a esta pandemia.

Referencias

1. Naqa, I. E., & Murphy, M. J. (2015). What Is Machine Learning? Machine Learning in Radiation Oncology, 3-11. https://doi.org/10.1007/978-3-319-18305-3_1.

2. Alemán, C. Y. (2019). Metodología para el aprendizaje ontológico semiautomático de dominio pedagógico. FCC-BUAP, 1-7.

3. Rettinger, A., Lösch, U., Tresp, V., D Amato, C., & Fanizzi, N. (2012). Mining the semantic web. Data Mining Knowledge Discovery, 24 (3), 613-662. https://doi.org/10.1007/s10618-012-0253-2.

4. Li, Q. (2020, 22 enero). An Outbreak of NCIP (2019-nCoV) Infection in China - Wuhan, Hubei Province, 2019−2020. China CDC Weekly, 2(5), 79-80.

5. Covid-19. (2020, 22 abril). Servicios de Salud. https://www.saludzac.gob.mx/home/index.php/covid-19/que-es/13-covid-19#:%7E:text=Es%20una%20enfermedad%20infecciosa%20causada,en%20diciembre%20de%202019.

6. Suárez, V., Suárez, M., Oros, S., & Ronquillo, E. (2020, mayo). Epidemiología de COVID-19 en México: del 27 de febrero al 30 de abril de 2020. Revista Clínica Española, 220(8), 463-471. https://doi.org/10.1016/j.rce.2020.05.007.

7. COVID-19 México. (2020). COVID - 19 Tablero México. https://datos.covid-19.conacyt.mx/.

8. Semáforo - COVID-19. (2020). Gobierno de México. https://coronavirus.gob.mx/semaforo/.

9. Morato, J. (2009). Information Retrieval. glossariumBITri. https://sites.google.com/site/glosariobitrum/Home/recuperacion-de-informacion.

10. Pinto, M. (2018). Búsqueda y Recuperación de Información. Electronic Content Management Skills. http://www.mariapinto.es/e-coms/busqueda-y-recuperacion-de-informacion/.

11. Chowdhury, G. G. (2003). Natural language processing. Annual review of information science and technology, 37 (1), 51-89.

12. Manning, C. D. & Schütze, H. (1999). Foundations of statistical natural language processing. Cambridge, MA, USA: MIT Press.

13. Smith, B. (2004). Ontology and Information Systems. Stanford.

14. Mahesh, K. (1996). Ontology development for machine translation: Ideology and methodology.

15. Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum. - Comput. Stud., 43, 907-928.

16. Weigand, H. (1997). A multilingual ontology-based lexicon for news filtering-the TREVI project. In Proceedings of the IJCAI Workshop on Multilingual Ontologies-Nagoya.

17. Faria, C., & Girardi, R. (2014). A domain-independent process for automatic ontology population from text. Science of Computer Programming, 95, Part 1, 26-43. Special Issue on Systems Development by Means of Semantic Technologies.

18. Heijst, V. & Guarino, N. (1997). Understanding, building, and using ontologies - a commentary to using explicit ontologies in kbs development, by van heijst, schreiber, and wielinga.

19. Sánchez López, S. E. (2007). Modelo de indexación de formas en sistemas VIR basado en ontologías. PhD thesis, Escuela de ingeniería y Ciencias. Universidad de las Américas Puebla.

20. Miñambres G., F., & Zheng, Z. (2016). Herramienta de apoyo al diagnóstico basada en el análisis de historias clínicas. Facultad de Informática, Universidad Complutense de Madrid. https://eprints.ucm.es/44661/1/Memoria_TFG.pdf.

21. He, Y., Yu, H., Ong, E., & Wang, Y. (2020, junio). CIDO, a community-based ontology for coronavirus disease knowledge and data integration, sharing, and analysis. Scientific Data, 7(1), 1-5. https://doi.org/10.1038/s41597-020-0523-6.

22. Mitchell, R. (2018). Web Scraping with Python: Collecting More Data from the Modern Web (2nd ed. ed.). O'Reilly Media.

23. Hiremath, O. S. (2020, 16 septiembre). A Beginner's Guide to learn web scraping with python. Edureka. https://www.edureka.co/blog/web-scraping-with-python/.

24. What is Web Scraping? (2020). Scrapinghub. https://www.scrapinghub.com/what-is-web-scraping/.

25. What is Python? Executive Summary. (2020). Python. https://www.python.org/doc/essays/blurb/.

26. Materiales del entrenamiento de programación en Python. (2018). COVANTEC. https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/ ventajas_ desventajas.html.

27. Wieringa, J. (2012, diciembre). Intro to Beautiful Soup. Programming Historian. https://programminghistorian.org/en/lessons/intro-to-beautiful-soup.

28. Edpresso Team. (2020, agosto). What is Beautiful Soup? Educative: Interactive Courses for Software Developers. https://www.educative.io/edpresso/what-is-beautiful-soup.

29. Bases de Datos NoSQL vs SQL. (2020, septiembre). Graph Everywhere. https://www.grapheverywhere.com/nosql-vs-sql/.

30. Bases de datos más utilizadas por los desarrolladores. (2019, 26 febrero). Diarlu. https://www.diarlu.com/gestores-bases-datos/.

31. What is ElasticSearch? (2020, septiembre). Elastic. https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html.

32. Rouse, M. (2018, 11 enero). MariaDB. SearchDataManagement. https://searchdatamanagement.techtarget.com/definition/MariaDB.

33. About MariaDB Server. (2020, 1 septiembre). MariaDB.org. https://mariadb.org/about/.

34. ¿Qué es big data? (2020). Oracle México. https://www.oracle.com/mx/big-data/what-is-big-data.html#link1.

35. Las tres V del Big Data: todo un reto por su volumen, variedad y velocidad. (2020, septiembre). UNIR. https://www.unir.net/ingenieria/revista/3-v-big-data/#:%7E:text=Las%20tres%20V%20del%20Big%20Data%20se%20refiere%20a%20los,adem%C3%A1s%20de%20sus%20principales%20retos.

36. Elshawi, R., Sakr, S., Talia, D., & Trunfio, P. (2018, mayo). Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service. Big data Research, 14, 1-11. https://doi.org/10.1016/j.bdr.2018.04.004.

37. What is Machine Learning? (2020, mayo). Expert.ai. https://www.expert.ai/blog/machine-learning-definition/.

38. Cloud Education, I. B. M. (2020, julio). Machine Learning. IBM. https://www.ibm.com/cloud/learn/machine-learning.

39. Mena, R. H., Velasco-Hernández, J. X., Mantilla-Beniers, N. B., Carranco-Sapiéns, G. A., Bernet, L., Boyer, D., & Pérez-Castillo, I. (2020, mayo). Using posterior predictive distributions to analyse epidemic models: COVID-19 in Mexico City. arXiv, 1-6.

40. Jeon, J., Baruah, G., Sarabadani, S., & Palanica, A. (2020, mayo). Identification of risk factors and symptoms of SARS-CoV-2 (COVID-19) using biomedical literature and social media data: Integrative and Consensus study. Journal of Medical Internet Research, 3-22. https://doi.org/10.2196/20509.

41. Wang, S., Zha, Y., Li, W., Wu, Q., Li, X., Niu, M., Wang, M., Qiu, X., Li, H., Yu, H., Gong, W., Bai, Y., Li, L., Zhu, Y., Wang, L., & Tian, J. (2020). A fully automatic deep learning system for COVID-19 diagnostic and prognostic analysis. European Respiratory, 56(2), 1-11. https://doi.org/10.1183/13993003.00775-2020.

42. Barrientos-Gutiérrez, T., lpuche-Aranda, C., Lazcano-Ponce, E., Pérez-Ferrer, C., & Rivera-Dommarco, J. (2020). La salud pública en la primera ola: una agenda para la cooperación ante COVID-19. Salud Publica Mex., 62, 598-606. https://doi.org/10.21149/11606.

43. Wu, G., Yang, P., Xie, Y., et al. (2020, agosto). Development of a clinical decision support system for severity risk prediction and triage of COVID-19 patients at hospital admission: an international multicentre study. European Respiratory, 56(2), 1-11. https://doi.org/10.1183/13993003.01104-2020.

44. Torrealba-Rodríguez, O., Conde-Gutiérrez, R. A., & Hernández-Javier, A. L. (2020, septiembre). Modeling and prediction of COVID-19 in Mexico applying mathematical and

computational models. Chaos, Solitons & Fractals, 138, 1-8. https://doi.org/10.1016/j.chaos.2020.109946.

45. Hurtado-Duarte, A. M., Rojas-Varela, R., García-Benítez, M., Hernández-Moralesa, A., Sotelo-Robledo, R., Gómez-Penagos, J., & Pensado-Piedra, L. (2020). Hallazgos tomográficos en afectación pulmonar por COVID-19, experiencia inicial en el Instituto Nacional de Enfermedades Respiratorias Ismael Cosío Villegas, Ciudad de México. NCT-Neumología y Cirugía de Tórax, 79(2), 71-77. https://doi.org/10.35366/94630.

46. Jani, B. D., Pell, J. P., McGagh, D., Liyanage, H., & Kelly, D. (202-08). Recording COVID-19 consultations: review of symptoms, risk factors, and proposed SNOMED CT terms. BJGP Open, 20, 1-10. https://doi.org/10.3399/bjgpopen20X101125.

47. Huang, K., Geller, J., Halper, M., Perl, Y., & Yu, J. (2009, junio). Using WordNet synonym substitution to enhance UMLS source integration. Artif Intell Med., 46(2), 97-109. https://doi.org/10.1016/j.artmed.2008.11.008.

48. Bello-Chavolla, O. Y., Bahena-López, J. P., Antonio-Villa, N. E., Vargas-Vázquez, A., & González-Díaz, A. et al., (2020, agosto). Predicting Mortality Due to SARS-CoV-2: A Mechanistic Score Relating Obesity and Diabetes to COVID-19 Outcomes in Mexico. The Journal of Clinical Endocrinology & Metabolism, 105(8), 2752–2761. https://doi.org/10.1210/clinem/dgaa346.

49. Hernández-Garduño, E. (2020, junio). Obesity is the comorbidity more strongly associated for Covid-19 in Mexico. A case-control study. Obes Res Clin Pract, 14(4), 375–379. https://doi.org/10.1016/j.orcp.2020.06.001.

50. Parra-Bracamonte, G. M., López-Villalobos, N., & Parra-Bracamonte, F. E. (2020, diciembre). Clinical characteristics and risk factors for mortality of patients with COVID-19 in a large data set from Mexico. Annals of Epidemiology, 52, 93–98. https://doi.org/10.1016/j.annepidem.2020.08.005.

51. Muhammad, L. J., Algehyne, E. A., Usman, S. S., Ahmad, A., Chakraborty, C., & Mohammed, I. A. (2020, 27 noviembre). Supervised Machine Learning Models for Prediction of COVID-19 Infection using Epidemiology Dataset. SN Computer Science, 2, 11. https://doi.org/10.1007/s42979-020-00394-7.

52. Hung, M., Lauren, E., Hon, E., & Birmingham, W. et al. (2020). Social Network Analysis of COVID-19 Sentiments: Application of Artificial Intelligence. Journal of Medical Internet Research, 22(8), 13. https://doi.org/10.2196/22590.

53. Hernández-Galdamez, D., González-Block, M., Romo-Dueñas, D., & Lima-Morales, R. et al. (2020, octubre). Increased Risk of Hospitalization and Death in Patients with COVID-19 and Pre-existing Noncommunicable Diseases and Modifiable Risk Factors in Mexico. Archives of Medical Research, 51(7), 683–689. https://doi.org/10.1016/j.arcmed.2020.07.003.

54. Weisstein, E. (2021, 18 mayo). SIR Model - from Wolfram MathWorld. Wolfram MathWorld. https://mathworld.wolfram.com/SIRModel.html.

55. SIR - A Model for Epidemiology. (2021). Systems Sciences. http://systems-sciences.uni-graz.at/etextbook/sw2/sir.html.

56. Just, W., & Terman, D. (2013). SIR Model - an overview | ScienceDirect Topics. ScienceDirect. https://www.sciencedirect.com/topics/mathematics/sir-model.

57. Smith, D., & Moore, L. (2016, 22 septiembre). The SIR Model for Spread of Disease - The Differential Equation Model. Mathematical Association of America. https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model.

58. Alimadadi, A., Aryal, S., Manandhar, I., Munroe, P. B., Joe, B., & Cheng, X. (2020). Artificial intelligence and machine learning to fight COVID-19. Physiological genomics, 52(4), 200–202. https://doi.org/10.1152/physiolgenomics.00029.2020.

59. Brinati, D., Campagner, A., Ferrari, D., Locatelli, M., Banfi, G., & Cabitza, F. (2020). Detection of COVID-19 Infection from Routine Blood Exams with Machine Learning: A Feasibility Study. Journal of medical systems, 44(8), 135. https://doi.org/10.1007/s10916-020-01597-4.

60. Izquierdo, J. L., Ancochea, J., Savana COVID-19 Research Group, & Soriano, J. B. (2020). Clinical Characteristics and Prognostic Factors for Intensive Care Unit Admission of Patients With COVID-19: Retrospective Study Using Machine Learning and Natural Language Processing. Journal of medical Internet research, 22(10), e21801. https://doi.org/10.2196/21801.

61. Bresfelean, V. P. (2007, junio). Analysis and Predictions on Students Behavior Using Decision Trees in Weka Environment. 29th International Conference on Information Technology Interfaces, 51–56. https://doi.org/10.1109/ITI.2007.4283743.

62. Gómez, A., & Verjel, A. (2015, julio). Minería de datos aplicada a la demanda del transporte aéreo en Ocaña, Norte de Santander. Tecnura, 19(45), 101–103. https://doi.org/10.14483/udistrital.jour.tecnura.2015.3.a08.

63. Hall, M., Frank, E., Holmes, G., Pfahringer, B., & Reutemann, P. (2009, junio). The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1), 10–18. https://doi.org/10.1145/1656274.1656278.